Systematic Approximation of Multi-Scale Feynman Integrals with Two-Loop Applications

---

Dissertation

zur

Erlangung der naturwissenschaftlichen Doktorwürde

(Dr. sc. nat.)

vorgelegt der

Mathematisch-naturwissenschaftlichen Fakultät

der

Universität Zürich

von

Daniel Paul Andrew Hulme

aus

Grossbritannien

Promotionskommission

Prof. Dr. Thomas Gehrmann (Vorsitz und Leitung der Dissertation)
Prof. Dr. Massimiliano Grazzini
Prof. Dr. Gino Isidori
Prof. Dr. Stefano Pozzorini

Zürich, 2019

This thesis is based on the author's work conducted at the University of Zürich. Parts of this work have already been published in Refs. [1].

Articles

[1] Borowka, Sophia and Gehrmann, Thomas and Hulme, Daniel
"Systematic approximation of multi-scale Feynman integrals",
Journal of High Energy Physics **08** (2018) 111,
`arXiv:1804.06824 [hep-ph]`

Zusammenfassung

Ein automatisiertes PYTHON-Programm, basierend auf einem Algorithmus (geschrieben in MATHEMATICA) für die parallelisierbare, systematische Abschätzung von Zwei-Schleifen-Feynman-Integralen mit mehreren Skalen, TAYINT, wird präsentiert. Das Programm erzeugt algebraische Ausdrücke als Funktionen der in den Feynman-Integeralen vorhandenen kinematischen Parameter und Massenskalen und ermöglicht so eine schnelle numerische Auswertung der Integrale mit einem akkuraten und präzisen Ergebnis. Die Abschätzungen sind in allen kinematischen Bereichen, sowohl unter als auch über Schwellenwerten, und, im Prinzip, bis zu beliebiger Ordnung im dimensionalen Regulator gültig. Die Abschätzung des Integranden wird mittels einer Taylorentwicklung in den Integrationsparametern erreicht. Um sicher zu gehen, dass die Entwicklung schnell genug konvergiert um eine gute Präzision der numerischen Ausdrücken zu garantieren, wurde der konzeptuelle TAYINT-Algorithmus entwickelt. Um die Anwendbarkeit des Algorithmus auf nichtplanare und elliptische Zwei-Schleifen-Feynman-Integrale zu erweitern, wurde ein Verfahren eingeführt, um das Schwellenverhalten numerisch zu lokalisieren. Die algebraische Abschätzung von elliptischen und nicht-planaren Feynman-Integralen ist schwieriger, weshalb zusätzliche Schritte eingeführt wurden, um die Anwendbarkeit des konzeptuellen Algorithmus zu stärken, mit dem finalen TAYINT-Algorithmus als Ergebnis. Der finale Algorithmus wurde dann als automatisiertes PYTHON-Programm umgesetzt. Um seine Anwendung zu demonstrieren, wird das Programm auf ausgewählte Zwei-Schleifen-Drei- und Vierpunkt-Integrale mit einer internen Massenskala angewendet, welche in den Zwei-Schleifen-Amplituden für die Higgs+Jet-Produktion auftauchen. Sie umfassen elliptische, nicht-planare und divergente Integrale. Um die Genauigkeit und Möglichkeit zur kinematischen Verallgemeinerung der algebraischen TAYINT-Abschätzungen zu zeigen, werden dann die sich daraus ergebenden numerischen Abschätzungen präsentiert.

Abstract

An automated program in PYTHON implementing an algorithm (written in MATHEMATICA) for the parallelisable, systematic approximation of multi-scale two-loop Feynman integrals, TAYINT, is presented. The program produces algebraic expressions as functions of the kinematical parameters and mass scales appearing in the Feynman integrals, allowing for fast numerical evaluation that leads to accurate and precise numerical expressions. The approximations are valid in all kinematical regions, both above and below thresholds and up to, in principle, arbitrary orders in the dimensional regulator. The approximation of the integrand is achieved by means of a Taylor expansion in the parameters of integration. In order to ensure that this converges quickly enough to guarantee both accuracy and precision in the numerical approximations, the conceptual TAYINT algorithm was developed. To bring non-planar and elliptic two-loop Feynman integrals within the algorithm's capability, the ability to numerically locate threshold-like behaviour was added. Elliptic and non-planar Feynman integrals are the most difficult class of Feynman integrals to approximate algebraically, so additional steps were included to strengthen the rigour of the conceptual algorithm, leading to the final TAYINT algorithm. This final algorithm was then promoted to an automated PYTHON program. To demonstrate its scope, the program is applied to selected two-loop three-point and four-point integrals, with an internal mass scale, that appear in the two-loop amplitudes for Higgs+jet production. These include elliptic, non-planar and divergent integrals. To demonstrate the accuracy, precision and kinematic generalisability of the algebraic TAYINT approximations, the resultant numerical approximations are then presented.

Contents

1 Introduction

1.1 Background to this Thesis

Feynman integrals [2] are a fundamental constituent of perturbative calculations in theoretical particle physics and many techniques have been developed to calculate them. Going beyond one loop, the calculation of multi-scale, multi-loop integrals is still a limiting factor in the theoretical predictions of hard processes.

The use of differential equations [3–10] and expression of the resulting integrals in terms of generalised polylogarithms [11–20] has proven most successful in the past and has led to a plethora of analytically-available results [21–34], leading in turn to important phenomenological predictions. With the presence of internal massive lines or particular non-planar graphs, elliptic structures appear which cannot be expressed in terms of polylogarithms alone. While progress is being made towards a description of these [35–41], the numerical approach using sector decomposition [42–45] in publicly-available programs [46–55] has become increasingly viable. This is demonstrated by its successful phenomenological applications up to two-loop, five-point, four-scale processes [34,56–60, 60–64]. Still, the numerical evaluation suffers from long evaluation times or is limited in accuracy. More often than not, a tuning of the integration parameters is needed to allow for a rapidly-converging, precise and accurate result at arbitrary kinematic points.

To shorten the evaluation times, the results for Feynman integrals must be algebraic in the kinematic parameters. These include Mandelstam invariants, external and internal particle masses. Moreover, the algebraic representation must always yield a precise numerical result in general, at *any* kinematic point. Then the evaluation at each kinematic point takes just as long as the time needed for the insertion of their numerical values. Algebraic approximations can be obtained if the integrands are Taylor expanded in the Feynman parameters, the integration is then over a polynomial. However, in their initial form Feynman integrands are not at all suited to a Taylor expansion in their integration parameters and it is of no avail reducing the complexity of the Feynman integral if it means losing the precision of the result.

1.1.1 Overall Aim of this Thesis

The aim of the research described in this thesis is to demonstrate that it is possible to reduce a Feynman integral to a polynomial integral whilst still retaining the complexity of

the integrand in coefficients that factor out of the integrals, ultimately leading to an accurate, precise and *algebraic* approximation. The challenge of this project was therefore to find a way to transfer the complexity of two-loop Feynman integrands from their integration parameters into the coefficients of a Taylor expansion in those parameters, such that the integrated expansion is a precise and kinematically algebraic approximation while the integration is straightforward to perform. To ensure that an approximation of this form is indeed rapidly converging, each integrand must be manipulated before any Taylor expansion is performed, so that it is in a form optimised for such an approximation. It is this manipulation that leads to the complexity of the integrand being encapsulated by the coefficients of the Taylor expansion and that requires the understanding of the topics and development of the program presented in the main body of this thesis.

This thesis presents an automated Python program, TayInt, to obtain such approximations and thus find a compromise between analytic and numerical approaches. TayInt contains a rigorous systematic algorithm to approximate algebraically any integral expressible in terms of Feynman parameters and generate an algebraic integral library which can be instantaneously evaluated.

The physical foundations provided by the Standard Model of particle physics are described in Chapter 2 and the relevant theory underpinning the TayInt method, concerning complex analysis and loop integrals, is covered in Chapters 3 and 4. The techniques of Quasi-Finite Basis reduction [65, 66] and Sector Decomposition [42–45] (used as preparatory steps in the TayInt program) are introduced and explored in Chapters 5 and 6. The theory behind the *threshold singularities* that constitute the major obstacle to developing the program are introduced in Chapter 7 and the reason why they present an obstacle is described in Chapter 8. The idea for approximating two-loop Feynman integrals algebraically in over-threshold kinematic regions is also given in this chapter. This idea is then built into a first algorithm for approximating two-loop Feynman integrals algebraically: the conceptual TayInt algorithm. This is described and illustrated in Chapter 9. The method of locating the thresholds of two-loop Feynman integrals is illustrated in Chapter 10 and the final TayInt algorithm for generating algebraic representations of highly complicated two-loop Feynman integrals in general is presented in Chapter 11. In this chapter, full details of the final algorithm are given alongside illustrative examples to demonstrate the rationale behind it. In Chapter 12 the automation of the algorithm into a Python program is discussed and a users' manual provided. The systematic TayInt approximations for a variety of two-loop integrals are collected and discussed in Chapter 13 and conclusions are drawn in Chapter 14.

TayInt was developed by observing the contours of the varying levels of evidence built up over extensive analysis of a variety of test cases to generate numerical approximations which are not only algebraic in the kinematic scales but can also be evaluated at arbitrary kinematic points so that they are both precise and accurate. The final TayInt algorithm was designed to analyse the structure of the Feynman integral from the perspective of the impending calculation as deeply as possible. Its structure makes maximal use of all the available evidence in order to exclude the unsuitable representations of the Feynman integral. This is done by a large enough margin to manipulate two-loop four-

point Feynman integrals up to the elliptic level such that they can be approximated algebraically.

The structure of the conceptual and final TayInt algorithms are illustrated in Tables 1.1, 1.2 below. The steps U1 and U2 in Table 1.1 use methods already developed, [42–45], [65, 66]. Steps BT1-2 and OT1-6 consist of entirely original work developed for the project described in this thesis. In Table 1.2 only step U1 relies on pre-existing work and the remaining steps are original. The labels of the steps in these tables will be typeset in bold as this thesis proceeds. But before that, the necessary mathematical and physical foundations to understand them must be presented, starting with the Standard Model of particle physics, in the following chapter.

Table 1.1: Summary of the individual steps of the conceptual TayInt algorithm.

U1: reduce the Feynman Integral to a quasi-finite basis	
U2: perform a sector decomposition on the finite integrals in the basis	
Below threshold	Over threshold
BT1: $t_j \to y_j$	OT1: $t_j \to \theta_j$, generate $\mathcal{K}$
BT2: Taylor expand the integrand and integrate	OT2: find optimum $\Theta_{o(0),\ldots,o(J-1)}$
	OT3: perform one-fold integrations
	OT4: post-integration, find optimum $\Theta_{o(0),\ldots,o(J-2)}$
	OT5: determine partition $\mathcal{P}_j$
	OT6: Taylor expand and integrate

Table 1.2: Summary of the individual steps of the final TayInt algorithm.

U1: perform a sector decomposition on the finite integrals in the basis			
U2: locate all the thresholds of the Feynman integral			
Below threshold		Over threshold	
BT1:	$t_j \to y_j$	**OT1**:	generate partition sets, full and partial contour configurations, kinematic training and cross-validation sets
BT2:	Taylor expand the integrand and integrate	**OT2**:	find partition:plain ratios for the full and partial contours
		OT3:	Choose optimal full and partial contour by negative exclusion, backup with positive selection
		OT4:	Perform kinematic cross validation
		OT5:	Assess the optimal full and partial contours and choose between them
		OT6:	Generate the uniform partition ratios on the chosen contours
		OT7:	Assess the accuracy of the chosen contours to decide if a high uniform partitioning can be used for all subsectors
		OT8:	If not, use the accuracy and improvement assessment to decide if a low or high uniform partitioning is appropriate
		OT9:	If not, analyse the subsectors on a per-variable basis and find the optimal varied partitioning
		OT10:	Taylor expand and integrate

1.2 Assumptions

Unless stated otherwise, the following assumptions are used throughout this thesis.

1. Natural units are used, so $\hbar = c = \epsilon_0 = 1$.

2. Einstein summation convention: repeated indices are summed over, for example $\bar{\psi}_f(i\not{\partial} - m_f)\psi_f(x)$ means $\sum_f \bar{\psi}_f(i\not{\partial} - m_f)\psi_f(x)$ where $f = u, d, c, s, t, b$.

3. Momenta are defined in four-dimensional Minkowski space with metric signature $\{+, -, -, -\}$. External momenta are always labelled with p_i and internal (loop) momenta with k_α where i runs over the number of external legs and α over the number of loops.

4. Loop integrals are considered exclusively within the scheme of conventional dimensional regularisation.

5. Within dimensional regularisation, the measure of loop integration is always taken to be $\int d^D k_\alpha / i\pi^{D/2}$.

6. The Feynman slash notation is used to simplify Dirac algebra, so that $\not{k} = \gamma^\mu k_\mu$.

7. As the conformal mappings used by TayInt are exclusively of the linear fractional type, the terms conformal mapping and linear fractional mapping will be used interchangeably except in Chapter 3.

2 | The Physical Origin of Loop Integrals

2.1 Introduction

As stated in Chapter 1, the goal of this thesis is to develop an automated PYTHON program capable of generating a library of algebraic approximations for two-loop four-point Feynman integrals up to the elliptic level, which can be quickly evaluated to produce accurate, precise numerical results at any kinematic point. However, before presenting the work which led to the accomplishment of this goal, its realisation must first be given meaning by introducing the physical theory within which this research is of relevance, namely, theoretical particle physics.

2.2 The Outline of Particle Physics

Particle physics is concerned with predicting the total probability of an event involving fundamental particles happening, [67]. This is known as the *total cross section* for that event, denoted by σ. Consider a scattering process between particles in which an initial asymptotic state $|i\rangle$ with total four-momentum p_i^μ evolves into an asymptotic final state $|f\rangle$ possessing total four-momentum p_f^μ by means of exclusively local interactions,

$$|i\rangle \xrightarrow{\text{local}} |f\rangle.$$

The relationship between these initial and final states as the system evolves is encapsulated by a quantity known as the *Scattering-matrix* (often abbreviated as *S-matrix*). For the generic scattering process described above, the S-matrix elements read:

$$\langle f|S|i\rangle = \underbrace{\delta_{fi}}_{\text{unscattered}} + \underbrace{i(2\pi)^4\delta\left(p_f^\mu - p_i^\mu\right)\mathcal{M}_{fi}}_{\text{scattering interaction}} . \tag{2.1}$$

The total cross section for the scattering process can then be mathematically expressed by taking the modulus squared of the interaction part of the scattering-matrix elements and integrating over the phase-space volume occupied by the final state of the system, $|f\rangle$,

$$\sigma = \int d\sigma \propto \int d\phi_f |\mathcal{M}_{fi}|^2 , \tag{2.2}$$

with the proportionality being turned into equality by incorporating an overall constant pertinent to the kinematics of the process, known as the *flux factor*. However, since experiments measure the final state of scattering processes at different kinematic points in the phase-space volume of the final state, theoretical predictions must be made which are also differential in the kinematic invariants. This is known as the *differential cross section*, $d\sigma$.

2.3 The Details of Particle Physics

In order to apply this outline and compute the cross section for a specific process, say, Higgs-plus-jet, it is necessary to know how to mathematically describe the Higgs boson and understand the quarks and gluons that give rise to a jet, as well as all the interactions between them and then a method for calculating these interactions mathematically must be developed. In summary, to move from the basic outline of differential cross sections to quantifying fundamental processes at experimental precision, the following developments are needed:

1. D1: provide the theoretical description of nature on the fundamental level,
2. D2: develop a mathematical formalism to make calculations within this fundamental theoretical arena provided by D1.

Development D1 was concretely established in 2012 when the discovery of the Higgs boson [68, 69] cemented the Standard Model of particle physics [70] as the definitive description of matter and its interactions on a fundamental level, whist development D2 was set in motion by the giants of Quantum Field Theory [2], including Feynman, Schwinger, Dyson and others. Given the extremely challenging nature of making precise predictions for processes involving fundamental particles, many steps have been taken to find a viable route to producing them; introducing a perturbative expansion, representing it diagrammatically, reducing the loop integrals to Feynman integrals and finding methods to compute Feynman integrals. The latter step is still a matter of current research, with which the work of this thesis is concerned, by presenting a novel approach to computing Feynman integrals as algebraic approximations. The formal simplicity of the Standard Model and the difficulty in making precision calculations within it is represented by number of pages devoted to each in this thesis, 12 and 154 respectively.

2.3.1 Colouring the Outline in (D1): the Standard Model of Particle Physics

The mathematical development of the Standard Model is presented in Table 2.1. The theory itself was finalised between 1970 and 1973 following the independent creation of the electro-weak theory by the work of Glashow in 1961 [71], Weinberg in 1967 [72] and Salam in 1968 [73] and the observation of the first evidence for quarks in 1969 [74, 75]. However, it is not the mathematical elegance of the Model that has led to it retaining the prefix Standard but rather its consistent experimental success. In 1974 the charm

Table 2.1: The Mathematical Development of the Standard Model of particle physics (this is not isomorphic to its chronological development). Entries in italics possess fermionic (half-integer) statistics and those which are underlined are massive. The respective typeset converses have bosonic (integer) statistics and are massless. The Higgs boson theoretically completes the Standard Model by breaking the unified electroweak sector $SU(2)_{\mathrm{L}} \times U(1)_{\mathrm{Y}}$ to the electromagnetic sector $U(1)_{\mathrm{em}}$ and in so doing, provides the explanation for the origin of the experimentally observed masses of the gauge bosons W^+, W^-, and Z. The experimental observation of the Higgs boson in 2012 established the Stanard Model as the definitive theory for use in predicting the scattering of fundamental particles.

Accumulated Theory	**Added Bosons**	**Feature Added**	**Matter Influenced**	**Physics Effect**	**Symmetry Development**
QED	Photon (γ)	Electro-magnetic force	*quarks*, *leptons*	Conserves electric charge, e	$U(1)_{\mathrm{em}}$
Electro-Weak	W^+, W^-, Z	Weak force	*quarks*, *leptons*	Conserves e, weak isospin	$SU(2)_{\mathrm{L}} \times U(1)_{\mathrm{Y}}$
QCD+Electro-Weak	Gluon (A^a_μ)	Strong force	*quarks*	Conserves colour charge	$SU(3)_{\mathrm{C}} \times SU(2)_{\mathrm{L}}$ $\times U(1)_{\mathrm{Y}}$
Standard Model	Higgs boson (H^0)	Higgs mechanism	*quarks*, *leptons*	Gives W^+, W^-, Z mass. Yukawa couplings.	$SU(2)_{\mathrm{L}} \times U(1)_{\mathrm{Y}}$ $\rightarrow U(1)_{\mathrm{em}}$

quark was observed [76, 77], then the bottom quark in 1977 [78], the W and Z bosons in 1983 [79] and the top quark in 1995 [80, 81]. In 2000, the tau neutrino was observed at Fermilab [82], completing the components necessary for the Standard Model. The only remaining puzzle was how to explain the masses of the W and Z bosons without contradicting the $SU(2)_{\mathrm{L}} \times U(1)_{\mathrm{Y}}$ symmetry of the electroweak sector that provided the correct structure for the observed particles in nature. A theoretical explanation had been provided in 1964 by a trinity of independent groups: Brout and Englert [83]; Higgs [84–86]; Guralnik, Hagen and Kibble [87]. However, this explanation necessitated the existence of a massive, spin-0 particle termed "the Higgs Boson", the discovery of which in 2012 finally cemented the Standard Model as the accepted theory of fundamental particles and their interactions.

2.3.2 Sharpening the Pencils (D2): The Perturbative Expansion

Concurrently with the development of the Standard Model (D1), the theoretical physics community was also developing the means to make predictions for scattering processes (D2) within each of its quantum field theories. However, even in the most mathematically straightforward of these, scalar field theories, analytically computing the S-matrix (Eq. (2.1)) from general principles is near-impossible, with the only feasible approach being a *perturbative expansion*. Returning to the idea of a generic scattering process, this scattering can be modelled mathematically by quantifying the exchanges of mediating particles (such as photons and gluons) governing the local interactions with a *coupling constant*, depending on the theory. In QED this coupling is in fact the fine structure constant,

$$\alpha_{\text{em}} = \frac{e^2}{4\pi} \, . \tag{2.3}$$

If this is a small number, the S-matrix can be very well represented by the first terms of a power series in the coupling constant, termed a perturbative expansion. However, it is a well-known feature of quantum field theories that the coupling strength actually changes as the energy scale at which the measurement is performed varies. For example, in Quantum Electrodynamics (QED), the value of α_{em} rises as more momentum Q^2 is transferred in the interaction, illustrated by:

$$\alpha_{\text{em}}(Q^2 \approx 0) \approx \frac{1}{137} \, , \tag{2.4}$$

$$\alpha_{\text{em}}(Q^2 \approx m_W^2) \approx \frac{1}{128} \, , \tag{2.5}$$

denoting by m_W the invariant mass of the W boson, which is about 80 GeV. In light of this running of the coupling, the earlier statement concerning the feasibility of a perturbative expansion as a tool for S-matrix computations must be rephrased:

> Provided that the coupling constant of a given theory is small within the experimentally accessible range of energies, the S-matrix computations necessary to make viably precise predictions for scattering processes within that theory can be performed perturbatively.

Therefore, at the heart of the challenge of making S-matrix computations there now sits a perturbative series and so a generic differential cross section for a process with coupling strength α consists of a series of the form:

$$d\sigma = \underbrace{d\sigma^{(0)}}_{\text{LO}} + \underbrace{\alpha d\sigma^{(1)}}_{\text{NLO}} + \underbrace{\alpha^2 d\sigma^{(2)}}_{\text{NNLO}} + \underbrace{\alpha^3 d\sigma^{(3)}}_{\text{N}^3\text{LO}} + \mathcal{O}(\alpha^4) \, . \tag{2.6}$$

Each order of this series corresponds to groups of *Feynman diagrams*, whose mathematical meaning is translated by using the *Feynman rules*, obtained using a standard recipe from the Lagrangian density of the relevant theory (QED, QCD etc). This process will be addressed in the case of QCD in the following section.

2.4 Perturbative QCD

This thesis is concerned with finding a novel algorithm for computing Feynman integrals in an accurate, precise and kinematically generalisable way which are chiefly of relevance to diagrams that arise within Quantum Chromodynamics. This part of the Standard Model will now be discussed in detail and the idea of the perturbative expansion illustrated within it.

2.4.1 The Lagrangian of QCD and its Feynman Rules

At the beginning of any quantum field theory sits the action [2],

$$\mathcal{S} = \int d^4x \mathcal{L}\left(\phi_i(x), \partial_\mu \phi_i(x)\right) , \tag{2.7}$$

the integral over the four dimensions of space-time, $d^4x = dtd^3x$, of the Lagrangian density, $\mathcal{L}$, which encapsulates the relativistic dynamics of the quantum fields, through its dependence on both them $\phi_i(x)$ and their derivatives $\partial_\mu \phi_i(x)$. This can be decoded into the equations of motion for the field theory by minimising the action with functional calculus,

$$\delta \mathcal{S} = 0 , \tag{2.8}$$

whose solutions yield the allowed values of the fields themselves.

The Quark Lagrangian

In the case of QCD, the goal is to construct a quantum field theory for *quarks* by writing down and mathematically developing a Lagrangian density that is consistent with their experimental description. As quarks are fermionic matter that come in six flavours, $f = u, d, c, s, t, b$, a good basis for the QCD Lagrangian density is the free Dirac Lagrangian well known to predict the behaviour of fermions,

$$\mathcal{L}_f(x) = \bar{\psi}_f(i\not{\partial} - m_f)\psi_f(x) , \tag{2.9}$$

$$\not{\partial} \equiv \gamma^\mu \partial_\mu . \tag{2.10}$$

However, unlike leptons, quarks require an additional quantum number if they are to comply with the Pauli exclusion principle, which is termed *colour*, $i = r, g, b$ and so the free Dirac Lagrangian evolves into the form:

$$\mathcal{L}_f(x) = \bar{\psi}^i_f(i\not{\partial} - m_f)\delta^{ij}\psi^j_f(x) . \tag{2.11}$$

Motivation for Gluons

In order to gain a deeper insight into the meaning of this colour-upgraded Lagrangian density, it is instructive to note that it possesses a global $SU(N_c)$ symmetry, where

$N_c = 3$, the number of colours. Explicitly and suppressing the flavour indices f for brevity, the Lagrangian density is invariant under transformations of the form:

$$\psi_i(x) \to U_{ij}\psi_j(x), \tag{2.12}$$

$$SU(N_c) \ni U_{ij} = \left[e^{-i\theta^a t^a}\right]_{ij} , \tag{2.13}$$

described by the group $SU(N_c)$ which is dependent on $N_c^2 - 1$ *real* parameters. The quark fields ψ_i belong to the "fundamental" representation of this group, in which the *generators* are the matrices t^a_{ij} where $a = 1, ..., N_c^2 - 1$. To ensure that this symmetry leads to a conserved quantity in accordance with Noether's theorem, known as "gauging" the symmetry, local invariance under $SU(N_c)$ is required, $U_{ij} \to U_{ij}(x)$, so that:

$$\psi_i(x) \to U_{ij}(x)\psi_j(x) , \tag{2.14}$$

$$SU(N_c) \ni U_{ij}(x) = \left[e^{-i\theta^a(x) t^a}\right]_{ij} . \tag{2.15}$$

In order to comply with this demand of local gauge invariance the Lagrangian must be modified by promoting the partial derivative to a "covariant" derivative D_μ which acts as an operator according to:

$$D_\mu U(x) = U(x) D_\mu , \tag{2.16}$$

$$D_\mu \equiv \partial_\mu + i g_s A_\mu(x) , \tag{2.17}$$

$$A_\mu(x) \to U(x) A_\mu U^\dagger(x) + \frac{i}{g_s} \left(\partial_\mu U(x)\right) U^\dagger(x) . \tag{2.18}$$

Using Eq. (2.15) reveals that $i \left(\partial_\mu U(x)\right) U^\dagger(x) = \partial_\mu \theta^a(x) t^a$ and so for the addition in Eq. 2.18 to make sense $A_\mu(x)$ must be a linear combination of the generators in their fundamental representation, t^a, thus:

$$A_\mu(x) = A_\mu(x)^a t^a , \tag{2.19}$$

$$\implies N_c^2 - 1 \text{ "gluons" } A^a_\mu(x) . \tag{2.20}$$

Understanding Gluons

To understand how gluons transform, the generators t^a must first be studied further. They form a "Lie algebra" of the group $SU(N_c)$, defined by:

$$[t^a, t^b] = i f^{abc} t^c , \tag{2.21}$$

$$\mathrm{Tr}[t^a t^b] = T_F \delta^{ab} , \tag{2.22}$$

where f^{abc} is a totally antisymmetric tensor of real numbers satisfying the Jacobi identity [88] and $T_F = \frac{1}{2}$, where the F denotes the fact that the t^a belong to the fundamental representation. Using Eqs. (2.18) and (2.21), the behaviour of $A^a_\mu(x)$ under an infinitesimal gauge transformation (θ small) can be uncovered,

$$A^a_\mu(x) t^a \to A^a_\mu t^a + \frac{1}{g_s} \left[\partial_\mu \theta^a - g_s f^{abc} A^b_\mu \theta^c\right] t^a . \tag{2.23}$$

Having identified the gluons and their transformations within the budding QCD Lagrangian density, the next step is to give them dynamics, by constructing the commutator of covariant derivatives, termed the "field strength" $F_{\mu\nu}$, defined as:

$$F_{\mu\nu} \equiv -\frac{i}{g_s}[D_\mu, D_\nu] = \partial_\mu A_\nu - \partial_\nu A_\mu + i g_s [A_\mu, A_\nu] \,, \tag{2.24}$$

which, using Eq. (2.21) can be simplified to;

$$F_{\mu\nu} = \partial_\mu A_\nu - \partial_\nu A_\mu - g_s f^{abc} A^b_\mu A^c_\nu \,, \tag{2.25}$$

and transforms as

$$F_{\mu\nu} \to U(x) F_{\mu\nu}(x) U^\dagger(x) \ . \tag{2.26}$$

Thus, the field strength tensor is not locally gauge invariant, due to the self-interactions of gluons via colour. From Eq. (2.26) it is already apparent that the field strength tensor resides in the adjoint representation of the $SU(N_c)$ Lie group defined by Eq. (2.21), the fundamental representation of which housed the quarks. Due to the fact that the A_μ fields can be decomposed in terms of the generators t^a, detailed in Eq. (2.19), the behaviour of the field strength tensor under an infinitesimal gauge transformation (θ small) reveals that:

$$F^a_{\mu\nu} t^a \to F^a_{\mu\nu} t^a + f^{cba} t^a F^c_{\mu\nu} \theta^c = t^a (\delta^{ab} - i(-if^{cab})\theta^c) F^b_{\mu\nu} \,, \tag{2.27}$$

which suggests the definition $(T^c)_{ab} = -if^{cab} = if^{cba} = -if^{abc} = (T^a)_{bc}$. This finally unveils that the matrices $(T^a)_{bc}$ are the generators of the adjoint representation of $SU(N_c)$, defined by:

$$[T^a, T^b] = i f^{abc} T^c \,, \tag{2.28}$$

which follows from the Jacobi identity for f^{abc}. The field strength tensor's transformation may therefore be finalised as:

$$F^a_{\mu\nu} \to (\delta^{ab} - i((T^c)_{ab})\theta^c) F^b_{\mu\nu} \ . \tag{2.29}$$

The Gluon Lagrangian

The covariant derivative, written in the fundamental representation in Eq. (2.16), can thus be written in the adjoint representation now the form of its generators is known and reads:

$$D^{ab}_\mu = \partial_\mu \delta^{ab} + i g_s (T^c)^{ab} A^c_\mu \,, \tag{2.30}$$

which enables the expression of the transformation of the gluon fields (given in the fundamental representation in Eq. (2.23)) in the adjoint representation,

$$A^a_\mu \to A^a_\mu + \frac{1}{g_s} D^{ab}_\mu \theta^b \ . \tag{2.31}$$

This is independent of the representation of the quark field ψ_f. Calling on Eq. (2.25), a gauge invariant Lagrangian density for the gluon fields can at last be constructed,

$$\begin{aligned}\mathcal{L}_g &= -\frac{1}{2}\mathrm{Tr}[F_{\mu\nu}F^{\mu\nu}] = -\frac{1}{4}\left((F_{\mu\nu})_a(F^{\mu\nu})^a\right) \\ &= -\frac{1}{4}\left(\partial_\mu(A_\nu)^a - \partial_\nu(A_\mu)^a\right)\left(\partial^\mu(A^\nu)^a - \partial^\nu(A^\mu)^a\right) \\ &+ \frac{g_s}{2}f^{abc}\left(\partial_\mu(A_\nu)^a - \partial_\nu(A_\mu)^a\right)(A^\mu)^b(A^\nu)^c \\ &- \frac{g_s^2}{4}f^{abc}f^{ade}(A_\mu)^b(A_\nu)^c(A^\mu)^d(A^\nu)^e \ . \end{aligned} \tag{2.32}$$

This emphasises the inference made earlier that gluons self-interact.

The Classical QCD Lagrangian

The result of compounding the quark Lagrangian, Eq. (2.11) and the gluon Lagrangian, Eq. (2.32), is the *classical QCD Lagrangian*, which reads:

$$\mathcal{L}_f + \mathcal{L}_g = \bar{\psi}_f^i(i\partial\!\!\!/ - m_f)\delta^{ij}\psi_f^j(x) - \frac{1}{4}\left((F_{\mu\nu})_a(F^{\mu\nu})^a\right) \ . \tag{2.33}$$

Quantising the QCD Lagrangian: Gauge-Fixing and Fadeev-Popov Ghosts

Using the Lagrangian density in Eq. (2.32) to construct an action (Eq. (2.7)) and subsequently imposing $\delta S = 0$ under variations which vanish at boundaries (Eq. (2.8)) generates the classical equations of motion for QCD. However, the gauge invariance of the Classical QCD Lagrangian density means that these equations are linearly dependent and thus cannot be uniquely inverted. To facilitate inversion of the equations of motion a "gauge fixing" condition must be imposed to remove the multiple linearly dependant equations of motion,

$$G_F[A] = 0\,, \tag{2.34}$$

which can be solved to fix the value of θ^a. Hence the term

$$\mathcal{L}_{GF} = -\frac{1}{2\xi}\left(G_F[A]\right)^2 \ , \tag{2.35}$$

must be incorporated into the Lagrangian density given in Eq. 2.33. Herein, ξ is the gauge fixing parameter. For instance, in the Lorentz gauge $G_F[A] = \partial^\mu A_\mu^a = 0$. The introduction of this term means that the QCD Lagrangian density can be quantised using the path integral formalism, as the integration is now carried out only over those fields A_μ^a which are gauge independent. However, the gauge-fixing $A_\mu^a = A_\mu^a(\theta^b)$ enforces a particular value of θ^a and so gives rise to a Jacobian determinant in the path integral, which obstructs the establishment of a perturbative expansion.

In order to construct a perturbative expansion using the path integral formalism and hence a *predictive* quantum field theory the integrand must be functionally differentiable, the realisation of which is prevented by the determinant. However, with the help

of Grassmann (anti-commuting) variables η^a and η^b, a functional determinant can be converted into a form similar to that of the standard integrand of a path integral, using the Berezin integral [2]:

$$\det\left(\frac{\partial f(x)}{\partial \omega(y)}\right) = \int \mathcal{D}\bar{\eta^a}\mathcal{D}\eta^b e^{i\int d^4x d^4y \bar{\eta^a}(x)\left[\det\left(\frac{\partial f(x)}{\partial \omega(y)}\right)\right]\eta^b(y)}\,, \tag{2.36}$$

which blends into the QCD path integral allowing a perturbative expansion to be subsequently constructed. The new spin-0 anti-commuting fields $\bar{\eta}^a$ and η^b are known as the "ghost" fields and introduce the Fadeev-Popov term into the QCD Lagrangian,

$$\mathcal{L}_{FP} = (\partial^\mu \bar{\eta^a}) D_\mu^{ab} \eta^b\,, \tag{2.37}$$

necessary for theory to admit a perturbative expansion. The gauge-fixing term enables the quantisation of the QCD Lagrangian density, the Fadeev-Popov terms ensures that it can be used for calculations.

The Complete and Quantised QCD Lagrangian

In summary, the QCD Lagrangian density is built as follows:

1. $\mathcal{L}_f$: as quarks are fermions which possess colour, the first piece of the Lagrangian density is the free Dirac Lagrangian density with colour incorporated.
2. $\mathcal{L}_g$: the $SU(N_c)$ symmetry associated with colour leads to a mathematical understanding of the gluon, the carrier of the colour force, which has dynamics described by a Lagrangian density consisting of a trace of non-Abelian field strength tensors.
3. $\mathcal{L}_{GF}$: to quantise this quark-gluon Lagrangian density the gluon fields must be assigned a particular gauge.
4. $\mathcal{L}_{FP}$: in order to use the gauge-fixed Lagrangian density for perturbative calculations, anti-commuting spin-0 fields known as ghosts must ultimately be introduced.

In its final form when choosing the Lorentz gauge, the quantised QCD Lagrangian density reads:

$$\begin{aligned}\mathcal{L}_{QCD} &= \mathcal{L}_f + \mathcal{L}_g + \mathcal{L}_{GF} + \mathcal{L}_{FP} \\ &= \bar{\psi}_f^i (i\partial\!\!\!/ - m_f)\delta^{ij}\psi_f^j - \frac{1}{4}\left((F_{\mu\nu})_a (F^{\mu\nu})^a\right) - \frac{1}{2\xi}\left(\partial^\mu A_\mu^a\right)^2 + (\partial^\mu \bar{\eta^a}) D_\mu^{ab}\eta^b\,.\end{aligned} \tag{2.38}$$

QCD Feynman Rules for the Propagators

The momentum-space Feynman rules corresponding to the Lagrangian density in Eq. (2.38) can then be obtained following a standard recipe. To reveal the form of the gluon propagator, integrating the quadratic terms in the A^a_μ (labelled $\mathcal{L}(A^2)$ by parts (remembering that the Lagrangian density sits inside an integral to form the action) and relabelling dummy indices yields:

$$\begin{aligned}\mathcal{L}(A^2) &= \frac{1}{2}\left(g_{\mu\nu}[\partial^2(A^\nu)^a](A^\mu)^a - [\partial_\mu\partial_\nu(A^\nu)^a](A^\mu)^a + \frac{1}{\xi}(\partial_\mu\partial^\nu A^a_\nu)(A^\mu)^a\right)\\ &= \frac{1}{2}(A^\mu)^a\left(\left[g_{\mu\nu}\partial^2 - \left(1-\frac{1}{\xi}\right)\partial_\mu\partial_\nu\right]\delta^{ab}(A^\nu)^b\right), \end{aligned} \tag{2.39}$$

which can be transferred to momentum space by making the replacement $\partial^\mu \to ik^\mu$;

$$\mathcal{L}(A^2) = \frac{1}{2}(A^\mu)^a\left(\left[-g_{\mu\nu}k^2 + \left(1-\frac{1}{\xi}\right)k_\mu k_\nu\right]\delta^{ab}\right)(A^\nu)^b, \tag{2.40}$$

and corresponds to the Green's function

$$G^{ab}_{\mu\nu} = \frac{1}{2}\left[-g_{\mu\nu}k^2 + \left(1-\frac{1}{\xi}\right)k_\mu k_\nu\right]\delta^{ab}. \tag{2.41}$$

The gluon propagator is obtained by finding the function $D^{\mu\nu}_{ab}$ solving the equation $G^{ab}_{\mu\nu}D^{\mu\nu}_{ab} = i$ which can be seen to be:

$$D^{\mu\nu}_{ab} = \frac{i}{k^2+i\epsilon}\left[-g^{\mu\nu} + (1-\xi)\,k_\mu k_\nu\right]\delta_{ab}. \tag{2.42}$$

The beauty of the Feynman rules is that they can be generated for each type of field by following the same recipe. This allows the rules for the fermion, gluon and ghost propagators to be represented by diagrams using only the knowledge of the Lagrangian. This is demonstrated in Table 2.2, in which all momenta are considered incoming.

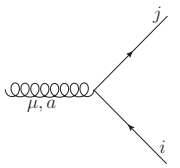
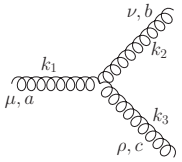
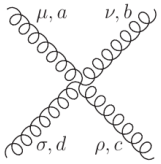
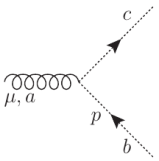
Table 2.2: The Feynman diagrams that pictorially describe the propagators in QCD.

Lagrangian Term	Feynman Rule	Feynman Diagram
$\frac{1}{2}(A^\mu)^a\left(G^{ab}_{\mu\nu}\right)(A^\nu)^b$	$\frac{i}{k^2+i\epsilon}\left[-g^{\mu\nu} + (1-\xi)\,k_\mu k_\nu\right]\delta_{ab}$	
$\bar{\psi}^i_f(i\not{\partial} - m_f)\delta^{ij}\psi^j_f$	$i\delta_{ij}\frac{\not{k}+m_f}{k^2-m_f^2+i\epsilon}$	
$(\partial_\mu\bar{\eta}_a)(D^{ab}_\mu\eta^b)$	$\frac{i\delta_{ab}}{k^2+i\epsilon}$	

QCD Feynman Rules for the Vertices

The Feynman rules for the QCD interaction vertices are given in Table 2.3. Each quark-gluon vertex corresponds to a factor of $-ig_s\gamma^\mu t^a_{ij}$ and each gluon self-interaction vertex corresponds to the momentum-space representation ($\partial^\mu \to ik^\mu$) of the coefficients of the cubic and quartic A^a_μ terms in $\mathcal{L}_g$, see Eq. (2.32). The flavour summation indices on the quark fields are again suppressed for brevity.

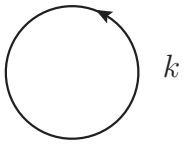
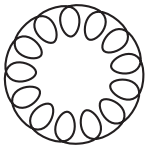
Table 2.3: The Feynman rules that pictorially encapsulate the QCD interaction vertices.

Lagrangian Term	Feynman Rule	Feynman Diagram
$-g_s\bar{\psi}_i\gamma^\mu A^a_\mu t^a_{ij}\psi_j$	$-ig_s\gamma^\mu t^a_{ij}$	
$\frac{g_s}{2}f^{abc}\left[\partial_\mu(A_\nu)^a - \partial_\nu(A_\mu)^a\right](A^\mu)^b(A^\nu)^c$	$-ig_s(if^{abc})$ $\times\left[(k_1-k_2)^\rho g^{\mu\nu}\right.$ $+(k_2-k_3)^\mu g^{\nu\rho}$ $\left.+(k_3-k_1)^\nu g^{\rho\mu}\right]$ $= g_s f^{abc}$ $\times\left[(k_1-k_2)^\rho g^{\mu\nu}\right.$ $+(k_2-k_3)^\mu g^{\nu\rho}$ $\left.+(k_3-k_1)^\nu g^{\rho\mu}\right]$	
$-\frac{g_s^2}{4}f^{abc}f^{ade}(A_\mu)^b(A_\nu)^c(A^\mu)^d(A^\nu)^e$	$-ig_s^2\left[f^{eac}f^{ebd}(g^{\mu\nu}g^{\rho\sigma}-g^{\mu\sigma}g^{\nu\rho})\right.$ $+f^{ead}f^{ebc}(g^{\mu\nu}g^{\rho\sigma}-g^{\mu\rho}g^{\nu\sigma})$ $\left.+f^{eab}f^{ecd}(g^{\mu\rho}g^{\nu\sigma}-g^{\mu\sigma}g^{\nu\rho})\right]$	
$-g_s f^{abc}(\partial_\mu\bar{\eta}^a)A^b_\mu\eta^c$	$ig_s k^\mu(-if^{abc}) = g_s f^{abc}k^\mu$	

QCD Feynman Rules for Loops

The Feynman rules provide a pictorial representation for particle physics processes within a given quantum field theory. However, in order to produce a precise prediction for a given process using the perturbative expansion, higher powers of the coupling constant g_s need to be incorporated. Mathematically, this means internal momenta must be incorporated, which must be integrated over. Pictorially, it corresponds to the inclusion of *loops*. For particles with fermionic (Grassmann) statistics, the anti-commuting nature of the Grassmann numbers means that loops come with a minus sign. The Feynman rules for loops in QCD are shown in Table 2.4.

Table 2.4: The Feynman diagrams that pictorially represent the inclusion of loops in QCD.

Feynman Rule	Feynman Diagram
$-\int \frac{d^4k}{(2\pi)^4}$	 k
$\int \frac{d^4k}{(2\pi)^4}$	 k
$-\int \frac{d^4k}{(2\pi)^4}$	 k

After translating all the mathematical building blocks for making perturbative predictions in QCD into Feynman diagrams, the symmetry factors that account for all permutations amongst identical particles must finally be added. These Feynman diagrams can then be used to construct pictorial representations for any QCD process.

2.4.2 Making Predictions with the QCD Lagrangian

In order to perturbatively calculate physical cross sections for processes involving quarks and gluons, the formalism of Feynman diagrams is used. Problems such as "factorisation" and "hadronisation" are not of relevance to this thesis, for reference, see [67]. At this point the assumptions of free quarks and gluons and their corresponding asymptotic

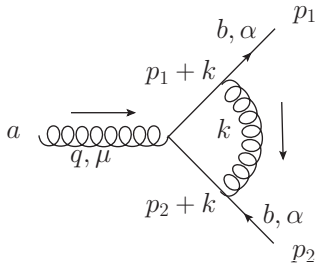


Figure 2.1: A Feynman diagram that contributes to the quark-gluon interaction at the one-loop level.

states are valid. Calculations at leading-order in the coupling constant g_s reproduce some features of the parton model but none of the interesting QCD effects are visible. This is because, in QCD higher-order corrections can be substantial owing to the large (relatively speaking, recall the values of the electromagnetic coupling constant given in Eqs. (2.4) and (2.5)) values over which the strong coupling constant runs,

$$\alpha_{\mathrm{s}} = \frac{g_s^2}{4\pi} \tag{2.43}$$

$$\alpha_{\mathrm{s}}(Q^2 \approx m_p^2) \approx 0.55\,, \tag{2.44}$$

$$\alpha_{\mathrm{s}}(Q^2 \approx m_Z^2) \approx 0.1\,, \tag{2.45}$$

where $m_p^2 \approx 938\mathrm{MeV}$, the mass of the proton and $m_Z^2 \approx 90$ GeV, the Z-boson invariant mass. To account for the interesting QCD effects, Feynman diagrams beyond leading-order must be included in the prediction.

The Emergence of Loop Integrals

For example, to mathematically describe the quark-gluon interaction for a particular quark with mass m_f at next-to-leading order (NLO), Feynman diagrams of the form in Fig. 2.1 must be included in the prediction of the S-matrix. Applying the Feynman rules in the Feynman gauge ($\xi = 1$) leads to an expression of the form:

$$\frac{g_s^3 t^a}{2N_c} \int \frac{d^4k}{(2\pi)^4} \frac{\gamma^\alpha(\not{p}_1 + \not{k} + m_f)\gamma_\mu(\not{p}_2 + \not{k} + m_f)\gamma_\alpha}{(k^2 + i\epsilon)\left[(p_1+k)^2 - m_f^2 + i\epsilon\right]\left[(p_2+k)^2 - m_f^2 + i\epsilon\right]}\,, \tag{2.46}$$

which must be evaluated. This is a *loop integral*. Thus, in order to add higher order predictions to the S-matrix, loop diagrams must be included and hence complicated loop integrals must be evaluated. The calculation of loop integrals is an in-depth subject but can be reduced to the problem of calculating Feynman integrals, which will be fully discussed in Chapter 4. Producing accurate, precise and kinematically generalisable results for Feynman integrals is the current obstacle to progress beyond the two-loop level in predicting differential cross sections in particle physics and shifting this obstacle

a little further forward is the goal of this thesis, described fully in Chapters 8-13. The meaning of the attempt to produce accurate, precise and kinematically generalisable approximations for Feynman integrals is now clear, it is synonymous with being able to make progress in computing differential cross sections precisely such that interesting physics effects are included.

Renormalisation

As will be discussed in Chapter 4, loop integrals are divergent and thus must be regularised if the integral is to be performed. However, as the results of the loop integrals are used to compute differential cross sections by insertion into S-matrices, the result *must* be finite. "Renormalisation" is the procedure through which all divergences that arise from Feynman diagrams at all orders in the perturbative expansion are shown to be an artefact of the definition of the parameters of the Lagrangian density, which can correspondingly be absorbed by a redefinition of the fields, masses and coupling constants.

2.5 Summary

Within this chapter the theoretical heart of predictions for differential cross sections in particle physics has been traced from the perturbative expansion to loop integrals. In Chapter 4 these integrals will be considered in detail, culminating in the generic expression for Feynman integrals, the current obstacle to progress in predicting differential cross sections at the two-loop level, which this thesis is concerned with attacking. However, prior to that, the necessary mathematics for understanding loop integrals and the TayInt program must be presented, which is the subject of the following chapter.

3 | Complex Analysis

3.1 Introduction

As explained in Chapter 1, converting the integrands of two-loop Feynman integrals into a form well suited for the application of a Taylor expansion requires considerable mathematical knowledge. The most important part of the mathematical literature that was relevant to developing the TayInt algorithm and program was that of *complex analysis*. This is because it allowed a contour of integration to be found that generates a representation of the subsectors of Feynman integrals well suited to a Taylor expansion in their parameters. The pre-required knowledge for performing this procedure will be provided in this chapter.

3.2 Outline of this Chapter

It was also explained in Chapter 1 that the work in this thesis aims to achieve the production of algebraic approximations for two-loop Feynman integrals in an *algorithmic* way. However, before such an algorithm can be developed, the raw mathematical method around which it is built must first be understood. This method consists of the following steps:

1. Understanding two-loop Feynman integrals mathematically;
2. Understanding how to move singularities in the integration parameters using *conformal mappings*;
3. Understanding how to convert the integrands into polynomials by means of a *Taylor expansion* in complex variables;
4. Understanding how to change the contour of integration and perform the integration in the complex space, such that singularities which depend on the external kinematic scales can also be avoided.

To understand two-loop Feynman integrands mathematically, the concepts of analyticity, conformality and singularities will be formally introduced in Section 3.4. In order to understand the relevant mathematical tools with which the final TayInt algorithm will be built, explanations of conformal mappings, Taylor expansions and complex integration

are provided in Sections 3.5, 3.6 and 3.7 respectively. But first, complex numbers must be introduced, in Section 3.3.

3.3 Why Complex Numbers are Useful

3.3.1 History

Complex numbers arise due to considering that equations may have *imaginary* solutions, for example $z^3 = 1$, which has solutions $z = 1, (-1)^{2/3}, (-1)^{4/3}$. This was promoted to a more formal branch of mathematics by Bernoulli, Euler, De Moivre and others, whose work led to a geometrical interpretation for complex numbers, $z = x + iy = r(\cos\vartheta + i\sin\vartheta) = re^{i(\vartheta+2m\pi)}$ where $r = \sqrt{x^2+y^2}$, the *modulus*, $\vartheta = arctan(y/x)$, the *argument* and m is an integer. The representation $z = e^{i\vartheta}$ is known as the Euler form and leads to the famous Euler formula, $e^{i\pi} = -1$ which Feynman went so far as to describe as the jewel of physics and engineering. Armed with this formalism, the equation $z^3 = 1$ can now be solved via $e^{i\vartheta} = e^{2m\pi i/3}$ so that $\vartheta = 0, 2\pi/3, 4\pi/3$ and hence $z = e^{2\pi i}, e^{2\pi i/3}, e^{4\pi i/3} = 1, (-1)^{2/3}, (-1)^{4/2}$.

The story of complex numbers began in 1543 when Cardano introduced complex numbers to solve cubic equations. Even though the equations had real roots, using complex numbers allowed the solutions to be found more easily. For example, using Cardano's formula, the equation

$$z^3 + 6z^2 + 9z + 3 = 0\,, \tag{3.1}$$

has solutions that can be obtained via complex methods and are in the form

$$z_1 = 2\cos\left(\frac{2\pi}{9}\right) - 2\,,$$
$$z_2 = 2\cos\left(\frac{8\pi}{9}\right) - 2\,,$$
$$z_3 = 2\sin\left(\frac{\pi}{18}\right) - 2\,.$$

3.3.2 Using Complex Methods to Solve Problems

Ever since, the trend of using complex methods to find real solutions more elegantly has continued and complex analysis has developed into a branch of mathematics in its own right, such is its power in proving theorems in mathematics and solving problems in physics. Further examples of problems which are solvable using complex methods include:

1. The prime number theorem: $\pi(n) \sim \frac{n}{\log(n)}$ which describes the asymptotic distribution of prime numbers, π over the positive integers n.

2. Stirling's formula for the asymptotic behaviour of the factorial function: $n! \sim \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$.
3. The Hardy-Ramanujan asymptotic partition formula: if n is a positive integer, then the number of unordered partitions of n, which are the unordered sequences of positive integers that sum to n, is asymptotically given by $p(n) \sim \frac{1}{4n\sqrt{3}} e^{\pi\sqrt{2n/3}}$.
4. Evaluating complicated real integrals, such as $\int_{-\infty}^{+\infty} \frac{\sin(ax)}{x}\, dx = \pi$.
5. Physics problems in hydrodynamics, heat conduction, electrostatics, etc.
6. Formulating the laws of the Universe on the most fundamental scales, as in Quantum Mechanics and Quantum Field Theory.
7. Conformal mappings, which are used in the TayInt code to improve the convergence of the Taylor expansions of Feynman integrands.

3.3.3 Why Complex Numbers are Important for TayInt

The fascinating things about all these examples is that they are not problems which involve complex numbers, nor are the solutions complex. However what has been repeatedly shown is that using complex techniques gives the mathematician or physicist a way to solve extremely difficult problems that do not seem to be related to complex numbers at all. This is still continuing at the present and in 2016 complex analysis was used to prove that the optimal densities for sphere packing in eight and 24 dimensions are $\frac{\pi^2}{384}$ and $\frac{\pi^2}{12!}$, [89]. When writing the TayInt program, for a long time it seemed impossible to find a representation of Feynman loop integrands that had convergent Taylor expansion in the Feynman parameters when the kinematic scales took on over-threshold values. This is not directly related to complex numbers but using complex analysis allowed new approaches to be taken, one of which ultimately solved the problem and is now the cornerstone of the automated program.

3.4 Mathematical Foundations

3.4.1 Motivation

It is first necessary to establish the notion of analyticity and conformality in the context of complex numbers, as, before beginning the endeavour to produce algebraic approximations for Feynman integrals up to the two-loop elliptic level, the integrals themselves must be rigorously understood, which requires a certain mathematical foundation.

3.4.2 Analyticity

Definition

In the mathematical literature the terms analytic, holomorphic and complex-differentiable are sometimes defined differently and then proved to be equivalent, however in this thesis

the terms will be used interchangeably. The object $f(z)$ is a function of the complex variable z if within a domain Ξ (a portion of the Argand diagram) there is a map that associates with every value of z at least one value of $f(z)$. Thus, $f(z)$ may be an arbitrary function consisting of some real and imaginary part, which are themselves functions of x and y. Denoting the real and imaginary parts of $f(z)$ by u and v respectively, allows $f(z)$ to be written as a function of x and y:

$$f(z) = u(x, y) + iv(x, y) \ . \tag{3.2}$$

A function $f(z)$ of a complex variable, single-valued in a domain Ξ, is analytic (complex-differentiable, holomorphic) at the point z in Ξ if:

$$f'(z) := \lim_{h \to 0} \frac{f(z+h) - f(z)}{h} \ , \tag{3.3}$$

exists uniquely. By unique, it is understood that the value of the limit is independent of the direction in Ξ along which $h \to 0$.

Example

For example, the function $f(z) = x^2 - y^2 + 2ixy$ is analytic for all values of z, as, rewriting $f(z) = z^2$, it is apparent that:

$$f'(z) := \lim_{h \to 0} \left[\frac{(z+h)^2 - z^2}{h} \right] = \lim_{h \to 0} (h + 2z) = 2z \ . \tag{3.4}$$

The function $f(z) = x^2 - y^2 + 2ixy$ is therefore analytic over the *entire* complex plane and thus is known as an *entire* function. On the other hand, the function $f(z) = 2y + ix$ is not analytic at any point in the complex plane, because, defining $h = h_x + ih_y$:

$$\frac{f(z+h) - f(z)}{h} = \frac{2y + 2h_y + ix + ih_x - 2y - ix}{h_x + ih_y} = \frac{2h_y + ih_x}{h_x + ih_y} \ , \tag{3.5}$$

so the limit of $h \to 0$ will change depending on its direction through the complex plane. If the limit is taken along a straight line of gradient m, $h_y = mh_x$, then:

$$\lim_{h \to 0} \left[\frac{f(z+h) - f(z)}{h} \right] = \lim_{h_x, h_y \to 0} \left[\frac{2m + i}{1 + im} \right] \ , \tag{3.6}$$

and so the derivative will be different for every value of m, regardless of what z is. Because of this, the function $f(z) = 2y + ix$ is not differentiable anywhere in the complex plane and hence is not an analytic function of z. However, a third possibility is that a function is analytic in a domain except at a finite number of points. These points are the *singularities* of $f(z)$. There are three types of singularities in complex analysis, as will be discussed later. For example, consider the function $f(z) = \frac{1}{(1-z)}$, which has the derivative:

$$f'(z) = \lim_{h \to 0} \left[\frac{f(z+h) - f(z)}{h} \right] = \lim_{h \to 0} \left[\frac{1}{(1 - z - h)(1 - z)} \right] = \frac{1}{(1-z)^2} \ . \tag{3.7}$$

This limit clearly exists, except at the point $z = 1$. Thus $f(z)$ is analytic, except at $z = 1$, which is known as a singularity.

Summary

Thus, in summary,

1. A function $f(z)$ is **analytic** within the domain Ξ consisting of the set of points z if the derivative can be said to uniquely exist at every point z in that domain.
2. An **entire** function is analytic over the entire complex plane.
3. A non-analytic point of a function is known as a **singularity**.

3.4.3 Formalising Analyticity

Motivation

The integrands of two-loop Feynman integrals are highly complicated mathematical objects and it is essential to more thoroughly understand what analyticity means in order to contemplate manipulating them such that they become suitable for a Taylor expansion, leading to algebraic approximations that in turn yield accurate, precise and kinematically generalisable numerical predictions.

The Cauchy-Riemann Equations

Having established the importance of analyticity, the next natural step is to formalise what it means to be analytic. Looking back at the examples given in Eqs. (3.3)-(3.7) suggests that the property of analyticity depends on the connection between the real and imaginary parts of $f(z)$. For a function $f(z)$ to be analytic, then

$$f'(z) = \lim_{h\to 0}\left[\frac{f(z+h)-f(z)}{h}\right], \tag{3.8}$$

must exist uniquely, which means that taking the limit along two different paths must yield the same result. Taking $f(z) = u(x,y) + iv(x,y)$ and $h = h_x + ih_y$, then:

$$f(z+h) = u(x+h_x, y+h_y) + iv(x+h_x, y+h_y)\,, \tag{3.9}$$

and the derivative is;

$$f'(z) = \lim_{h_x\to 0, h_y\to 0}\left[\frac{u(x+h_x,y+h_y)+iv(x+h_x,y+h_y)-u(x,y)-iv(x,y)}{h_x+ih_y}\right]. \tag{3.10}$$

If the function is to be analytic, then taking this limit along the real and imaginary axes must produce the same result. The limit along the real axis, $h_y = 0$, is given by:

$$f'_x(z) = \lim_{h_x\to 0}\left[\frac{u(x+h_x,y)-u(x,y)}{h_x} + i\frac{v(x+h_x,y)-v(x,y)}{h_x}\right] = \frac{\partial u}{\partial x} + i\frac{\partial v}{\partial x}\,, \tag{3.11}$$

and that along the imaginary axis, $h_x = 0$, amounts to:

$$f'_y(z) = \lim_{h_y \to 0} \left[\frac{u(x, y+h_y) - u(x,y)}{ih_y} + i\frac{v(x, y+h_y) - v(x,y)}{ih_y} \right] = -i\frac{\partial u}{\partial y} + \frac{\partial v}{\partial y} \ . \quad (3.12)$$

For $f(z)$ to be analytic, $f'_x(z) = f'_y(z)$. Equating the real and imaginary parts in Eqs. (3.11) and (3.12) leads to the necessary conditions for $f(z)$ to be analytic:

$$\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y} \,, \quad (3.13)$$

$$\frac{\partial v}{\partial x} = -\frac{\partial u}{\partial y} \ . \quad (3.14)$$

These are the *Cauchy-Riemann relations*.

Formal Definition of Analyticity

With these conditions, the concept of analyticity can be fully formalised. For a function of a complex variable, $f(z)$, to be analytic, the necessary and sufficient conditions are that the four partial derivatives $\frac{\partial u}{\partial x}, \frac{\partial v}{\partial y}, \frac{\partial v}{\partial x}, \frac{\partial u}{\partial y}$:

1. Must **exist**;
2. Must be **continuous**;
3. Must satisfy the **Cauchy-Riemann equations**.

For a more complete discussion of this, see for example [90, 91].

Analytic Continuation

One further and very important application of the concept of analyticity, highly relevant to theoretical particle physics, is that of *analytic continuation*. The principle of analytic continuation is contained in the following theorem [90]

Theorem 3.4.1 *Let $f_1(z)$ be an analytic function on the domain Ξ_1, which has one, and only one, overlapping subregion: Ξ_{sub} with the domain Ξ_2. Then, if there exists a function $f_2(z)$ which is analytic on Ξ_2 and coincides with $f_1(z)$ in Ξ_{sub}, then $f_1(z)$ and $f_2(z)$ are analytic continuations of one another.*

The above theorem asserts that provided Ξ_{sub} is not the empty set, then $f_2(z)$ is unique. A further implication is that $f_1(z)$ and $f_2(z)$ have *equal representations* in the overlapping subregion Ξ_{sub} but outside of this subregion, the two functions have different representations. An example of analytic continuation is provided by the Euler and Weierstrass

representations of the gamma function, [88], respectively

$$\Gamma_E(z) = \int_0^\infty dt\ t^{z-1}e^{-t},\ \mathrm{Re}[z] > 0 \tag{3.15}$$

$$\Gamma_W(z) = \left[\sum_{n=0}^{\infty} \frac{(-1)^n}{n!(n+z)}\right] + \int_1^\infty dt\ t^{z-1}\ e^{-t}\ . \tag{3.16}$$

The Euler representation is analytic in the domain $\mathrm{Re}[z] > 0$, which is Ξ_1 in this case. The Weierstrass representation is analytic everywhere except at the points $z = 0, -1, -2, -3, ...$, which is Ξ_2. As the two domains Ξ_1 and Ξ_2 clearly overlap, then the $\Gamma_W(z)$ is a unique analytic continuation of $\Gamma_E(z)$.

3.4.4 Conformality

The analyticity of a function can also be interpreted in a geometric way: an analytic function $f(z)$ is a local rotation and rescaling. This can be quickly understood by reference to the Argand diagram. Multiplication by i is equivalent to rotating through an angle of $\frac{\pi}{2}$ anti-clockwise. Multiplication by $2 + 3i$ corresponds to a rotation by an angle $\arctan(3/2)$ and a rescaling by a factor of $\sqrt{2^2 + 3^2} = \sqrt{13}$. If $\frac{df}{dz}|_{z=z_0} = f'(z_0)$ at $z = z_0$, then the infinitesimal change $df = f(z_0) + df - df$ results from rotating and rescaling dz by the argument and modulus of the derivative $f'(z_0)$. Now, consider two $\mathcal{C}^1$ curves (continuous first derivatives), $c_1(t)$ and $c_2(t)$. These curves are maps to regions Ξ of the complex space $\mathbb{C}$. At the point $t = t_0$, the angle between these two curves is ϑ. It is given by the normed inner product between the two tangent vectors to the curves at that point,

$$\cos\vartheta = \frac{\langle c_1'(t_0), c_2'(t_0)\rangle}{|c_1'(t_0)||c_2'(t_0)|}\ . \tag{3.17}$$

Given an analytic function $f(z) : \Xi \to \mathbb{C}$, then two more $\mathcal{C}^1$ curves can be generated by composition with f:

$$w_1(t) = f \circ c_1(t) = f(c_1(t))\,, \tag{3.18}$$

$$w_2(t) = f \circ c_2(t) = f(c_2(t))\,, \tag{3.19}$$

because f is analytic and thus infinitely many times smoothly differentiable, $\mathcal{C}^\infty$. Now using the chain rule,

$$w_1'(t) = f'(c_1(t))c_1'(t)\,, \tag{3.20}$$

$$w_2'(t) = f'(c_2(t))c_2'(t)\ . \tag{3.21}$$

The inner product of w_1 and w_2 is equal to:

$$\langle w_1'(t_0), w_2'(t_0)\rangle = \langle f'(c_1(t_0))c_1'(t_0), f'(c_2(t_0))c_2'(t_0)\rangle = \tag{3.22}$$

$$|f'(c_1(t_0))||f'(c_2(t_0))|\langle c_1'(t_0)\,, c_2'(t_0)\rangle\,, \tag{3.23}$$

from which it follows that the angle between the curves w_1 and w_2 at the point t_0 is given by the normed inner product of their tangent vectors at that point, namely:

$$\cos\varphi = \frac{\langle w_1, w_2\rangle}{|w_1||w_2|} = \frac{|f'(c_1(t_0))||f'(c_2(t_0))|\langle c_1'(t_0), c_2'(t_0)\rangle}{|f'(c_1(t_0))||f'(c_2(t_0))||c_1'(t_0)||c_2'(t_0)|} \tag{3.24}$$

$$= \frac{\langle c_1'(t_0), c_2'(t_0)\rangle}{|c_1'(t_0)||c_2'(t_0)|} = \cos\vartheta \ . \tag{3.25}$$

This shows that $\varphi = \vartheta$, provided that $f'(z) \neq 0$ (as otherwise φ will be undefined). Thus the composition of the curves with an analytic function leaves their orientation unchanged: the multiplication of the tangents to the original curves by the tangent to an analytic function preserves the angle between the curves at each point at which the function is analytic and the curves differentiable. Thus:

> Analytic functions $f(z)$ are **conformal**, as they preserve angles of orientation, provided that $f'(z) \neq 0$.

3.4.5 Singularities

Motivation

In order to make precise predictions for Feynman integrals that are kinematically generalisable, functions with many singularities need to be Taylor expanded. Consequently, in addition to a formal understanding of analyticity, a thorough understanding of the singularities of complex functions is also paramount before moving on, of which there exist three types.

Classification

1. Pole: a point $z = z_0$ which is singular is identified as a pole of order $0 < n \in \mathbb{Z}$ if and only if:

$$f(z) = \frac{g(z)}{(z - z_0)^n} \ , \tag{3.26}$$

with $g(z)$ being a non-zero analytic function at $z = z_0$. The most straightforward example is $f(z) = a(z - z_0)^{-n}$ where a is a non-vanishing complex constant.

2. Branch point: singularities of this type can best be illustrated by appealing to the example of the complex logarithm. Recalling the Euler form, $z = re^{i\vartheta}$, the complex logarithm takes the form

$$\log z = \log r + i\vartheta \ . \tag{3.27}$$

The real part of a complex logarithm is then extracted as:

$$\mathrm{Re}(\log z) = \log r = \log(\sqrt{x^2 + y^2}) \, , \tag{3.28}$$

which is a well-defined function except at the origin. The complex logarithm has the imaginary part:

$$\mathrm{Im}(\log z) = \vartheta = \arg[z] \ , \tag{3.29}$$

which is just the argument of the complex number. This is also undefined at the origin but is multi-valued everywhere else, since differences in argument of $2m\pi$ leave the complex number unchanged. To elaborate, every non-zero complex number has an infinite number of possible values for its argument, each separated by integer multiples of $2\pi i$, because $e^{2\pi i} = 1$. Therefore, there are also an infinite number of possible values for the imaginary part of the complex logarithm. For real numbers, $z = x$:

a) If $x > 0$, then $\log z = \log x$, as long as $\arg[x] = 0$. The other possibilities for the logarithm have an integer multiple of $2\pi i$ added, so ordinary real positive numbers also have logarithms that are complex.

b) If $x < 0$, then because $\log(-k) = \log\left(|k|e^{i\pi+2m\pi i}\right) = \log|k| + i\pi(2m+1)$, the logarithm is always complex regardless of the value chosen for the argument.

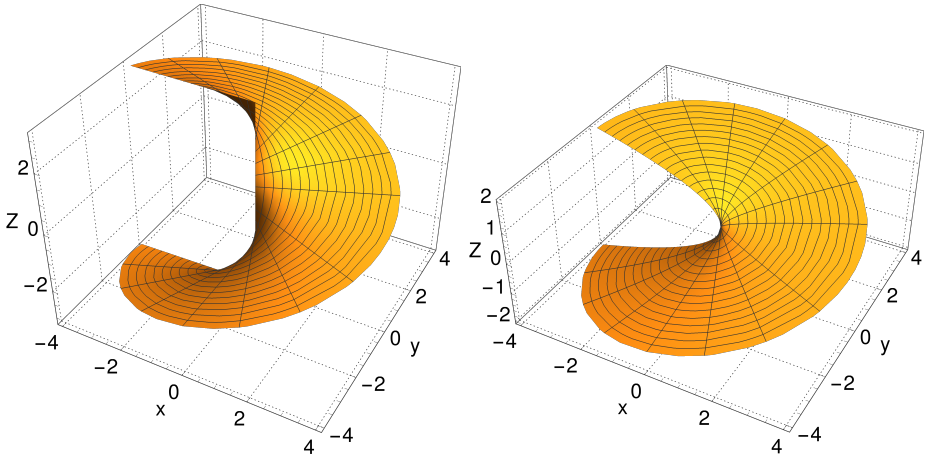
Each rotation of 2π around $z = 0$ changes the value of the complex logarithm, each of which is referred to as a *branch*. The central point $z = 0$ is known as the *branch point*, which is a type of singularity due to the infinite number of possible values the logarithm can take by rotating around it. The logarithmic branch point therefore has degree ∞. A similar branching effect takes place when taking roots of complex numbers. In the most elementary case, every non-zero complex number has two different square roots, $\pm\sqrt{z}$. Using the Euler form:

$$\sqrt{z} = \sqrt{r}e^{i\vartheta/2+2m\pi i/2} = \sqrt{r}e^{i\vartheta/2}e^{m\pi i} \tag{3.30}$$

$$= \begin{cases} \sqrt{r}e^{i\vartheta/2}, & \text{if } m \text{ is even} \\ -\sqrt{r}e^{i\vartheta/2}, & m \text{ is odd} \end{cases} , \tag{3.31}$$

demonstrating that the even and odd multiples of πi account for the two different values of the square root. Rotating the complex number z around the origin leads to *two* different values for the square root, first rotating from 0 to 2π changes $\sqrt{z}$ to $-\sqrt{z}$ and a further rotation of 2π returns the complex square root to $\sqrt{z}$. Hence this is a branch point of degree *two* and $z^{1/n}$ has a branch point of degree n. The branch points of the complex logarithm and the complex square root are illustrated in Fig. 3.1. Having illustrated the concept with examples, it only remains to make a formal statement concerning branch points. The function $z^a = e^{a\log z}$:

- Is analytic at $z = 0$ if $a \in \mathbb{Z}$ is an integer;
- Has an algebraic branch point of degree q at $z = 0$ if $a = p/q \in \mathbb{Q}$ is a rational, non-integer with $p \neq 0$ and $q \geq 2$ having no shared factors;
- Has a logarithmic branch point of infinite degree at $z = 0$ when $a \in \mathbb{C} \setminus \mathbb{Q}$ is not rational.



(a) (b)

Figure 3.1: The branches of: (a) the complex logarithm and (b) the complex square root.

Branch points are also termed *removable singularities* because they can be removed by restricting the function to a particular domain, for example by defining the logarithm on $\vartheta \in [0, 2\pi]$ the singularity is removed. As logarithms and square roots frequently appear in Feynman integrands, especially at higher orders in the perturbative expansion, understanding branch points is essential.

3. Essential singularity: if a singularity cannot be classified as either a pole or a branch point, then it is an *essential singularity*. An essential singularity is a pole of order ∞. The standard example is $e^{1/z}$ at $z = 0$.

In general, a function can have more than one kind of singularity, for example:

$$f(z) = \frac{e^{-1/(z-1)^2}}{(z^2+1)(z+2)^{2/3}}\,, \tag{3.32}$$

has order 1 (known as simple) poles at $z = \pm i$, a branch point of degree three at $z = -2$ and an essential singularity at $z = 1$.

3.4.6 Summary

In this section, the concepts of analyticity, conformality and non-analyticity have been introduced and formalised. These concepts are essential for understanding two-loop Feynman integrands. In the next sections, the mathematical concepts necessary to understand how to manipulate them are introduced, namely, those of conformal mappings, the Taylor expansion, and complex integration. Now that the concepts have been introduced which allows two-loop Feynman integrands to be formally understood, the mathematical tools that will be used to manipulate them must also be introduced, formalised and demonstrated. This is the subject of the subsequent sections in this chapter, beginning with the tool of conformal mappings.

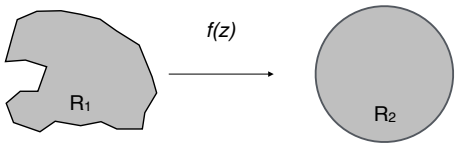


Figure 3.2: The analytic mapping between Ξ_1 and Ξ_2.

3.5 Conformal Mappings

3.5.1 Motivation

Conformality is not just an interesting property that provides a geometric interpretation of analyticity, it also provides an inspired way of changing the variables of a function so that it is more suitable for certain calculations. As two-loop Feynman integrands contain singular points in their Feynman parameters outside of their region of integration after performing an iterated sector decomposition, a particular class of conformal mappings, known as *linear fractional mappings* are considered to attempt to move these singularities as far away from the integration region as possible. To discuss this, the topic of treating complex analytic functions as mappings needs to be properly introduced.

3.5.2 Analytic Mappings

Complex analytic functions can also be seen as *conformal mappings* that transform a point $z = x + iy$, belonging to the domain $\Xi_1 \subset \mathbb{C}$, into $\zeta = \xi + i\eta$ on the image domain $\Xi_2 = f(\Xi_1) \subset \mathbb{C}$. This re-casting can be seen by writing the analytic function, $f(z)$, as:

$$u(x, y) + iv(x, y) = f(z) = \zeta = \xi + i\eta\,, \tag{3.33}$$

making manifest the link between each domain. As an unambiguous link between the two domains in required, a general requirement of the mapping in Eq. (3.33) is that it is *bijective*. This means that each point $\zeta \in \Xi_2$ can be uniquely traced to a point $z \in \Xi_1$. The inverse function $z = f^{-1}(\zeta)$ is then a sensibly defined map from Ξ_2 back to Ξ_1 and must also be analytic on Ξ_2. Requiring that the inverse also be analytic means that:

$$\frac{d}{d\zeta} f^{-1}(\zeta) = \frac{1}{f'(z)}\,, \tag{3.34}$$

and so $f'(z) \neq 0$ must hold everywhere for $f^{-1}(\zeta)$ to be analytic.

3.5.3 Linear Fractional Mappings

One of most important classes of mappings are the linear fractional mappings:

$$f(z) = \zeta = \frac{az + b}{cz + d}\,, \tag{3.35}$$

where a, b, c, d are complex constants, subject to the constraint $ad - bc \neq 0$. This constraint arises because if $ad - bc = 0$, then Eq. (3.35) becomes:

$$f(z) = \zeta = \frac{a}{c}\,, \tag{3.36}$$

which is trivial. An example of a linear fractional mapping is given by:

$$f(z) = \zeta = \frac{z-1}{z+1}\,. \tag{3.37}$$

This mapping is bijective and admits the inverse:

$$f^{-1}(z) = z = \frac{1+\zeta}{1-\zeta}\,, \tag{3.38}$$

which is analytic provided that $z \neq -1, \zeta \neq 1$, as:

$$f'(z) = \frac{2}{(z+1)^2}\,. \tag{3.39}$$

3.5.4 Relevance to TayInt

These mappings are so important because they enable a correspondence between infinite and constant values. The point $c \neq 0$ and $z = -d/c$ is mapped to $f(z) = \zeta = \infty$. Inversely, the point $z = \infty$ is mapped to the point $f(z) = \zeta = a/c$, which is a finite constant as long as $c \neq 0$. More formally, linear fractional mappings define bijective, analytic mappings from the *Riemann sphere*, $\mathbb{S} \equiv \mathbb{C} \cup \{\infty\}$ (the complex plane made contiguous with a point at infinity) to the complex space $\mathbb{C}$, and vice versa. As shown in Eq. (3.25), analytic functions are conformal and so the analytic mappings given in this section must all preserve angles of orientation. Linear fractional mappings are helpful for the TayInt program because they move certain points to infinity. When Taylor expanding an integrand, the convergence of the Taylor expansion is given by the distance to the nearest point of non-analyticity. Finding a linear fractional mapping that moves the non-analytic points in a domain an infinite distance away in the image domain means that the convergence of the Taylor expansion, and hence the precision of the result for the Feynman integral, is maximised. As the conformal mappings relevant for TayInt are linear fractional mappings, for the remainder of this thesis, when the term conformal mapping is used, this refers to a linear fractional type of mapping.

3.6 Complex Power Series

The goal of this thesis is to produce algebraic approximations for Feynman integrals which evaluate to yield precise, accurate numerical approximations at arbitrary kinematic points, without performing the integration exactly. To realise this goal it is necessary to find a way to use a Taylor expansion to simplify the integration whilst retaining the structure of the integral. The concepts of analyticity and non-analyticity have now

been formally introduced so that the properties of the Feynman integrands themselves can be rigorously understood. The next step is to fully understand the the means of approximating them. Thus, the Taylor expansion, beginning from the concept of a power series, will be introduced next.

3.6.1 Ordinary Power Series

Motivation

The object

$$P(x) = \sum_{i=0}^{\infty} a_i x^i = a_0 + a_1 x + a_2 x^2 + \dots , \tag{3.40}$$

is a power series, where the a_i are constant. Such representations are much used in physics because, for some non-vanishing $x_0 \in \mathbb{R}$ with bounded a_i, when there exists $M \geq 0$ such that $|a_i x_0^i| \leq M$, then for $|x| < |x_0|$, higher-order terms in the series can be dropped if $x_0 = 1$. This allows very complicated equations to be simplified considerably in the limit of small quantities and still match the required experimental precision, as was discussed in the section concerning the perturbative expansion of differential cross sections within the preceding Chapter 2. This can allow theoretical predictions that were impossible in full generality to be achieved.

Truncating Power Series

For example, the series in Eq. (3.40) can be truncated when x is small,

$$P(x) = \sum_{i=0}^{\infty} a_i x^i = a_0 + a_1 x + a_2 x^2 + \mathcal{O}(x^3) \approx a_0 + a_1 x + a_2 x^2 \ . \tag{3.41}$$

The symbols $\mathcal{O}$ and $\approx$ will be used extensively throughout this thesis and require formal definitions. They compare the way two functions behave when one of their shared variables approaches a limit. Denoting these two functions as $f(x)$, $g(x)$ and the limit as $x \to x_l$;

1. If $k \in \mathbb{R}$ such that $|f| \leq kg$ as x_l is approached then $f = \mathcal{O}(g)$.
2. If as $x \to x_l$ $f/g \to l$, $l \neq 0$, then $f \approx lg$. Writing $f \approx g$ means that $f/g \to 1$.

Quantifying Convergence

If a power series representation is to be used as an approximation of a function, then it is crucial that this power series represents the true function ever more precisely as more terms are retained in the series. That is, the power series must *converge*. There are many tests for determing whether or not a series converges (for a complete discussion of see [88]). If the series does not approach a well-defined limit as more terms are added, then it is said to be divergent, for example $1 + 2 + 3 + 4 + 5 + ...$ For the general

power series in Eq. (3.40), the D'Alembert ratio test [88] gives the condition for $P(x)$ to converge absolutely as:

$$\lim_{n\to\infty}\left|\frac{a_{n+1}}{a_n}x\right| = |x|\left|\frac{a_{n+1}}{a_n}\right| < 1\ . \tag{3.42}$$

The fact that the convergence of the series depends on the value of x defines a *region of convergence*: the range of values of x within which $P(x)$ converges. At the boundaries of this region, the series maybe convergent or divergent. To decide which it is, the boundary values of x must be inserted into the power series and the convergence of the resulting series assessed. For example, the series $P(x) = 1 + 2x + 4x^2 + 8x^3 + ...$ has $\lim_{n\to\infty}\left|\frac{2^{n+1}}{2^n}x\right| = |2x|$ so the power series is convergent for $|x| < 1/2$. The boundary points of this region of convergence are $1/2$ and $-1/2$. Inserting these into the power series yields $P(1/2) = 1 + 1 + 1 + 1 + ...$, which is clearly a divergent series and $P(-1/2) = 1 - 1 + 1 - 1 + ...$, which is oscillatory. Hence the series does not converge at either end point and so the region of convergence is $-1/2 < x < 1/2$.

Rules for Working with Power Series

In theoretical physics, it is not usually enough to simply understand the convergence of a power series, what happens when the power series is used in a calculation must also be properly understood, as will be seen in the chapters devoted to the development of the final TayInt algorithm (9-11). The following rules apply to both real and complex power series:

1. If the regions of convergence of $P_1(x)$ and $P_2(x)$ coincide then their sum, difference or product will converge in the mutually convergent region.

2. If the series $P_1(x)$ and $P_2(x)$ are everywhere convergent then the series obtained by inserting one into the other will also converge for all values of x. For example,

$$\sin x = x - \frac{x^3}{3!} + x^5 5! - ..., \tag{3.43}$$

$$e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + ...\,, \tag{3.44}$$

$$\implies e^{\sin x} = 1 + x + \frac{x^2}{2!} - \frac{3x^4}{4!} + ...\,, \tag{3.45}$$

   which also converges everywhere. On the other hand, if either series only has a limited region of convergence, then substituting one series into the other will not necessarily lead to a series that converges everywhere. If $P_1(x)$ is convergent everywhere but $P_2(x)$ has a limited region of convergence, then $P_2(P_1(x))$ may not be convergent, as $P_1(x)$ may not always fall within the region of convergence for $P_2(x)$.

3. If $P(x) = a_0 + a_1 x + a_2 x^2 + ...$ converges in the region $|x| < \lim_{n\to\infty} |a_n/a_{n+1}| \equiv k$ then the series

$$\frac{dP}{dx} = a_1 + 2a_2 x + 3a_3 x^2 + ..., \tag{3.46}$$

converges if

$$\left|\frac{(n+1)a_{n+1}x^n}{(n)a_n x^{n-1}}\right| = \left|\frac{(n+1)a_{n+1}x}{(n)a_n}\right| < 1 \tag{3.47}$$

$$\implies |x| < \lim_{n\to\infty}\left|\frac{na_n}{(n+1)a_{n+1}}\right| = \lim_{n\to\infty}\left|\frac{a_n}{a_{n+1}}\right| = k\,, \tag{3.48}$$

differentiating (3.46) term by term and using that $n/(n+1) \to 1$ as $n \to \infty$. The series

$$\int dx P(x) = a_0 x + a_1 \frac{x^2}{2} + a_2 \frac{x^3}{3} + ..., \tag{3.49}$$

will converge under the condition that:

$$\left|\frac{(n+1)a_{n+1}x^{n+2}}{(n+2)a_n x^{n+1}}\right| = \left|\frac{(n+1)a_{n+1}x}{(n+2)a_n}\right| < 1 \tag{3.50}$$

$$\implies |x| < \lim_{n\to\infty}\left|\frac{(n+2)a_n}{(n+1)a_{n+1}}\right| = \lim_{n\to\infty}\left|\frac{a_n}{a_{n+1}}\right| = k\,, \tag{3.51}$$

integrating term by term and using that $(n+2)/(n+1) \to 1$ as $n \to \infty$. This shows that

> Differentiating or integrating a *power series* does not change its region of convergence. However, the convergence behaviour at the boundaries of the region of convergence may change after differentiation or integration.

3.6.2 Taylor Series

Motivation

As discussed in the previous subsection, power series can be immensely useful for simplifying calculations that could not otherwise be performed and still obtaining results consistent with the required level of theoretical precision. In order to take advantage of this and associate a power series to a two-loop Feynman integrand by means of a specific, programmable formula, *Taylor's theorem* must be used. The particular power series generated by Taylor's theorem is known as a *Taylor series* or *Taylor expansion*.

Formalism

A single-valued and continuous function $f(x)$ can be written as:

$$\int_e^{e+h} dx\ f^{(1)}(x) = f(e+h) - f(e)\,, \tag{3.52}$$

where h is small, such that $e+h$ and e are in the same local neighbourhood. Rearranging this equation,

$$f(e+h) = f(e) + \int_e^{e+h} dx\ f^{(1)}(x)\,, \tag{3.53}$$

and then approximating $f^{(1)}(x)$ by $f^{(1)}(e)$, yields the *first approximation* to $f(e+h)$:

$$f(e+h) \approx f(e) + \int_e^{e+h} dx\ f^{(1)}(e) = f(e) + hf^{(1)}(e)\ . \tag{3.54}$$

Therefore, expanding a distance h around the point e, $x = e+h$, the first approximation to the function $f(x)$ is given by:

$$f(x) \approx f(e) + (x-e)f^{(1)}(e)\ . \tag{3.55}$$

Repeated differentiation of Eq. (3.54) yields:

$$f^{(1)}(x) \approx f^{(1)}(e) + (x-e)f^{(2)}(e)\,, \tag{3.56}$$

$$f^{(2)}(x) \approx f^{(2)}(e) + (x-e)f^{(3)}(e)\,, \tag{3.57}$$

$$\vdots$$

Inserting Eq. (3.56) into Eq. (3.53) gives rise to the *second approximation* to $f(e+h)$:

$$\begin{aligned} f(e+h) &\approx f(e) + \int_e^{e+h} dx\ f^{(1)}(e) + (x-e)f^{(2)}(e) \\ &\approx f(e) + hf^{(1)}(e) + \left[\frac{e^2}{2} + eh + \frac{h^2}{2} - e^2 - eh - \frac{e^2}{2} + e^2\right] f^{(2)}(e) \\ &\approx f(e) + hf^{(1)}(e) + h^2 f^{(2)}(e)\ . \end{aligned} \tag{3.58}$$

This procedure can be iterated ad infinitum (as long as the function is analytic) to generate increasingly higher-order approximations to the function. The $(n-1)^{\text{th}}$ approximation is:

$$f(e+h) \approx f(e) + hf^{(1)}(e) + \frac{h^2}{2!}f^{(2)}(e) + ... + \frac{h^{n-1}}{(n-1)!}f^{(n-1)}(e)\,, \tag{3.59}$$

where the *order* of the expansion is the highest power of h that appears in the Taylor series. Of course, Eq. (3.59) is not an exact replica of $f(x)$, it is an approximation. As such, there is an error associated to it, within which the true value of $f(x)$ lies. The error associated with a Taylor series up to order $(n-1)$ is of the order n, the next term in the series that is dropped and is given by:

$$T_n(h) = \frac{h^n}{n!}f^{(n)}(E)\,, \tag{3.60}$$

where $E \in [e, e+h]$.

Statement of Taylor's Theorem

Taylor's theorem states that the following equality holds:

$$f(e+h) = f(e) + hf^{(1)}(e) + \frac{h^2}{2!}f^{(2)}(e) + ... + \frac{h^{n-1}}{(n-1)!}f^{(n-1)}(e) + T_n(h)\,, \quad (3.61)$$

where $E \in [e, e+h]$. Thus, substituting $x = e+h$, $f(x)$ can be written as:

$$f(x) = f(e)+(x-e)f^{(1)}(e)+\frac{(x-e)^2}{2!}f^{(2)}(e)+...+\frac{(x-e)^{n-1}}{(n-1)!}f^{(n-1)}(e)+T_n(x)\,, \quad (3.62)$$

with the remainder:

$$T_n(x) = \frac{(x-e)^n}{n!}f^{(n)}(E),\ E \in [e, x]\ . \quad (3.63)$$

Relevance for TayInt

The formula (3.62) defines the *Taylor expansion* of the function $f(x)$ about the point $x = a$. The Taylor expansion is so important for the TayInt program that it forms half of the name. This is because TayInt was written to provide kinematically algebraic approximations to arbitrary Feynman integrals which are extremely difficult or even impossible to perform analytically. As the Feynman integrals cannot be analytically calculated, TayInt uses a Taylor expansion to transform the integrand into a polynomial that can be integrated over the Feynman parameters. Of course, a generic Taylor expansion of a two-loop Feynman integrand with several scales converges very poorly and is not precise enough to be a realistic prediction. However, that is what the TayInt program is for: turning a Feynman integrand into a function that *can* be precisely represented by a Taylor expansion. Once this is done, the Taylor expansion and integration are the final steps. Feynman integrands are functions of multiple variables, so a multi-variable version of Taylor's theorem is required. For an n-variable function $f(x_1, x_2, ..., x_n)$, it is more compact to use vector notation, $\vec{x} = (x_1, x_2, ..., x_n)^T$, such that the function is written as $f(\vec{x})$. Taylor's theorem then takes the form

$$f(\vec{x}) = f(\vec{x}_0) + \sum_i \frac{\partial f}{\partial x_i}\Delta x_i + \frac{1}{2!}\sum_i\sum_j \frac{\partial^2 f}{\partial x_i \partial x_j}\Delta x_i \Delta x_j + ...,\ \Delta x_i = x_i - x_{i_0}\ . \quad (3.64)$$

The partial derivatives are evaluated at the points $x_{i_0} = (x_{1_0}, x_{2_0}, ..., x_{3_0})$. Using $\nabla = (\partial/\partial x_1, ...\partial/\partial x_n) = \partial_i$, this can be further compressed,

$$f(\vec{x}) = \sum_{n=0}^{\infty} \frac{1}{n!}\left[(\Delta\vec{x}\cdot\nabla)^n f(\vec{x})\right]_{\vec{x}=\vec{x}_0}\ . \quad (3.65)$$

3.6.3 Complex Power Series

Motivation

In order to avoid the singularities of two-loop Feynman integrals which depend on the external kinematic scales (threshold singularities), as a first step the integrand needs to be mapped to the complex space. Therefore, the notion of Taylor series must also be extended to complex variables.

Extension to Complex Numbers

Applying Taylor's theorem to functions of complex variables leads to another definition of analyticity: a complex function $f(z)$ is analytic at the point $z_0 \in \mathbb{C}$ if its Taylor series expansion $\sum_{n=0}^{\infty} a_n(z-z_0)^n$ converges for all z close to z_0. To understand how the theory of power series in general and Taylor series in particular extend to functions of a complex variable, consider the case:

$$f(z) = \sum_{n=0}^{\infty} a_n z^n \,, \tag{3.66}$$

with z a complex variable and the a_n complex constants. The series $\sum_{n=0}^{\infty} a_n(z-z_0)^n$ can be obtained from Eq. (3.66) by performing the transformation $z \to (z-z_0)$. So, for brevity the series in Eq. (3.66) will be considered for the discussion here. Using $z = re^{\pi\theta}$, Eq. (3.66) becomes:

$$f(z) = \sum_{n=0}^{\infty} a_n r^n e^{in\theta} \,. \tag{3.67}$$

Quantifying Convergence

This series will converge absolutely if the positive real series:

$$f(z) = \sum_{n=0}^{\infty} |a_n| r^n \,, \tag{3.68}$$

is convergent. Thus, to assess the convergence of the complex series, it is only necessary to test the absolute convergence of a real series. To do this, any of the convergence tests discussed extensively in Chapter 4 of [88] can be used. Taking the Cauchy root test, the value of r within which the series will converge, termed the *radius of convergence*, is given by:

$$\frac{1}{r_c} = \lim_{n\to\infty} |a_n|^{1/n} \,, \tag{3.69}$$

the series in Eq. (3.66) converges absolutely if $|z| < r_c$, diverges if $|z| > r_c$. If $|z| = r_c$ further information is required.

Examples of Complex Power Series

This can be demonstrated by providing some intuitive examples of complex power series. The series:

$$\sum_{n=0}^{\infty} \frac{z^n}{n!}\,, \tag{3.70}$$

has $r_c = \infty$ since $\lim_{n\to\infty}(1/n!)^{1/n} = 0$ because $(n!)^{1/n} \sim n$ as $n \to \infty$. By the same logic, the series:

$$\sum_{n=0}^{\infty} z^n n!\,, \tag{3.71}$$

has $r_c = 0$. The series:

$$\sum_{n=0}^{\infty} \frac{z^n}{n}\,, \tag{3.72}$$

converges for $|z| < 1$ as $r_c = \lim_{n\to\infty}(1/n)^{1/n} = 1$. This final example is an excellent illustration of the statement that on the boundary of the circle of convergence, it is impossible to say whether or not the series converges or diverges without further analysis. With $z = 1$, the series reads:

$$1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \ldots\,, \tag{3.73}$$

which is divergent but with $z = -1$ the series becomes:

$$-1 + \frac{1}{2} - \frac{1}{3} + \frac{1}{4} - \ldots\,, \tag{3.74}$$

which is convergent and converges to $-\ln 2$.

Further Quantification of Convergence

The ratio test [88] can also be used to define the *radius of convergence*, as, according to this test:

$$\lim_{n\to\infty} \frac{|a_{n+1}||z|^{n+1}}{|a_n|z|^n}\,, \tag{3.75}$$

and so the radius of convergence reads:

$$\frac{1}{r_c} = \lim_{n\to\infty} \frac{|a_{n+1}|}{|a_n|}\,, \tag{3.76}$$

Applying this to the series in Eq. (3.71) yields:

$$\frac{1}{r_c} = \lim_{n\to\infty} \frac{n!}{(n+1)!} = \lim_{n\to\infty} \frac{1}{n+1} = 0\,, \tag{3.77}$$

and thus $r_c = \infty$ as found previously using Eq. (3.69). This analysis shows that, within the radius of convergence of a series, Taylor's theorem must hold and so the statement with which this section began can now be more formally stated: the Taylor series $\sum_{n=0}^{\infty} a_n(z - z_0)^n$ sums to an analytic function of z within the radius of convergence of the Taylor series.

Relevance for TayInt

One of the most important results in complex analysis for this thesis is that the radius of convergence for the Taylor series of an analytic function $f(z)$ can be found by inspection. Thus, there is no need for deciding which is the best convergence test to use.

> The radius of convergence, r_c, of a function with a Taylor series representation, $f(z) = \sum_{n=0}^{\infty} a_n(z - z_0)^n$ is equal to the distance from z_0 to the nearest *singularity* of $f(z)$, which is the nearest point of non-analyticity.

As discussed previously in Subsection 3.4.2, an *entire* function is analytic everywhere, thus has no singularities and so has $r_c = \infty$. Examples of entire functions include $e^z, \cos z, \sin z$. However, the function:

$$f(z) = \frac{1}{z^2 + 1} = \frac{1}{(z + i)(z - i)}\,, \tag{3.78}$$

contains singularities, at $z = \pm i$. So, the radius of convergence of its Taylor series is 1; the distance from $z_0 = 0$ to the nearest singularity. As is often the case, introducing complex variables resolves a point of confusion in the theory of real variables, namely; why the real function, $(1 + x^2)^{-1}$ is well-defined and analytic for all real x but is only convergent for $|x| < 1$. The complex singularities cripple convergence when $|x| > 1$. When expanding about a non-zero point, finding the distance to the singularities provides an easy way of finding the radius of convergence of the Taylor series. For example, expanding Eq. (3.78) about $z_0 = 1+2i$, the distances to the two singularities are $\sqrt{2}$ and $\sqrt{10}$. Thus the radius of convergence reduces to $\sqrt{2}$ if $z_0 = 0 \to 1 + 2i$.

An objective of the TayInt program is to provide precise results for Feynman integrals. As this precision is determined by the convergence of the Taylor expansion of the integrand, understanding how to optimise the radius of convergence is of crucial importance to the success of TayInt. One of the ways that the radius of convergence of the integrand's Taylor expansion is maximised is by using conformal mappings to move the singular points in the parameters of integration infinitely far away, as previously discussed in Subsection 3.5. Thus these two areas of complex analysis are unified in order to maximise the precision achievable by TayInt.

3.6.4 Demonstration that a Differentiable Complex Power Series is a Taylor Series

To conclude this section, recall that in Eq. (3.48) it was demonstrated that a convergent power series can be differentiated, with the resulting series having the same region of

convergence. Thus, a convergent series will have a derivative that is also a convergent series, thus is also differentiable to produce another convergent series, and so on. Hence,

> a convergent series is infinitely differentiable.

Taking the convergent series:

$$f(z) = a_0 + a_1(z-z_0) + a_2(z-z_0)^2 + a_3(z-z_0)^3 + ... = \sum_{n=0}^{\infty} a_n(z-z_0)^n\,, \quad (3.79)$$

and differentiating,

$$\begin{aligned} f^{(1)}(z) &= a_1 + 2a_2(z-z_0) + 3a_3(z-z_0)^2 + 4a_4(z-z_0)^3 + ... \\ &= \sum_{n=0}^{\infty}(n+1)a_{n+1}(z-z_0)^n\,, \quad (3.80) \\ f^{(2)}(z) &= 2a_2 + 6a_3(z-z_0) + 12a_4(z-z_0)^2 + 20a_5(z-z_0)^3 + ... \\ &= \sum_{n=0}^{\infty}(n+1)(n+2)a_{n+2}(z-z_0)^n\,, \quad (3.81) \end{aligned}$$

reveals that:

$$\begin{aligned} a_0 &= f(z_0)\,, \quad (3.82) \\ a_1 &= f^{(1)}(z_0)\,, \quad (3.83) \\ a_2 &= \frac{1}{2}f^{(2)}(z_0), \quad (3.84) \\ &\vdots \\ a_n &= \frac{f^{(n)}(z)}{n!}\,, \quad (3.85) \end{aligned}$$

which can be substituted into Eq. (3.79) to write the convergent power series as:

$$f(z) = \sum_{n=0}^{\infty}\frac{f^{(n)}(z)}{n!}(z-z_0)^n\,, \quad (3.86)$$

the Taylor series. This is not a full proof of Taylor's theorem for complex functions, which is given in Appendix **??**.

3.6.5 Summary

Now that Taylor's theorem has been introduced and generalised to the case of complex variables, the tool for converting a complex-mapped Feynman integrand into a polynomial has been understood. This will be seen to be important for the development of the final TayInt algorithm. As two-loop Feynman integrands are highly complicated objects with many integration variables and external kinematic scales, their Taylor expansion in complex variables will also be a complicated object. Thus, to ultimately integrate such a Taylor expansion, an in-depth understanding of the subject of complex integration is required, which will be presented in the next section.

3.7 Complex Integration

3.7.1 Motivation

Now that the tools of the conformal mapping and the Taylor expansion have been explained, the final tool needed for the TayInt program is that of complex integration: the Feynman integrals must ultimately be integrated. The crucial features of complex integration that are important for the development of the TayInt program are:

1. It allows multiple possible contours of integration, a feature of crucial importance for avoiding the threshold singularities of two-loop Feynman integrals.
2. A complex integral is independent of the specific parametrisation of the curve C that connects the end points of the integration interval. It is dependent on the orientation: reversing the paths direction corresponds to a change of sign in the result of the integral;
$$\int_{-C} dz\ f(z) = -\int_{C} dz\ f(z)\ . \tag{3.87}$$
3. Even if the end points of integration are the same, moving clockwise and anticlockwise around the path of integration will give different results if the path encloses a singularity.

These features will now be illustrated with an example.

3.7.2 Illustration

As a necessary beginning, rudimentary calculus teaches that the definite integral of a real function, $\int_a^b dt\ f(t)$ is computed along the interval $[a, b] \subset \mathbb{R}$. However when variables can be complex, integrals are computed along curves in the complex plane, just like the line integrals in real calculus. Thus there are more possible intervals of integration, as the complex space is two-dimensional. The route traversing the interval of integration is termed an integration *path*: a closed path is termed a *contour*. The corresponding *contour integral* is denoted as $\oint$. A good way to introduce the power of this construction is to compute complex integrals:

$$\int_C dx\ z^n\,, \tag{3.88}$$

with integrand $f(z) = z^n$ along a variety of different curves C, shown in Fig. 3.3.

Straight Line Segment

1. The first curve is the straight line $C = C_S$ that connects the points -1 and 1 by running along the real axis. This curve is parametrised as $z(t) = t$ for $-1 \leq t \leq 1$.

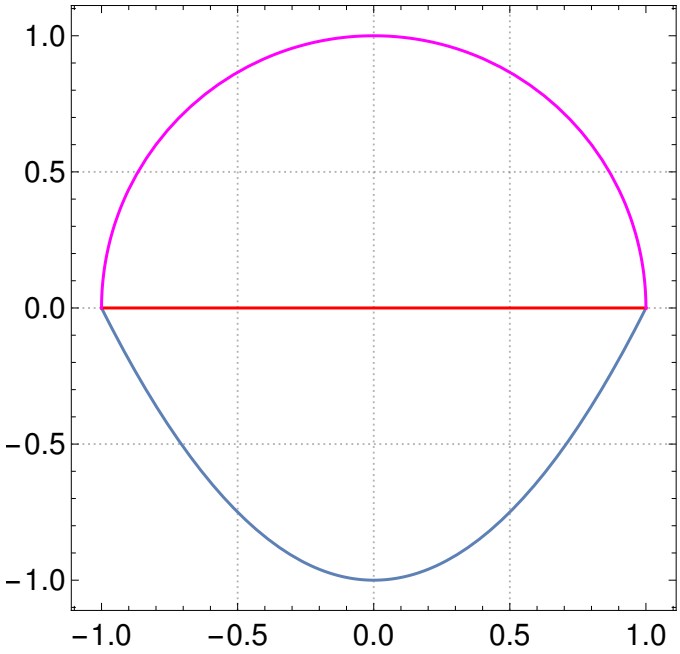


Figure 3.3: The integration curves:

- $z(t) = t$ for $-1 \leq t \leq 1$, used in Eq. (3.89), shown in red,
- $z(t) = t + i(t^2 - 1),\ -1 \leq t \leq 1$ used in Eq. (3.93), shown in blue,
- $z(t) = e^{it}$, $0 \leq t \leq \pi$ used in Eq. (3.94), shown in magenta.

It is shown in red in Fig. 3.3. The integral in Eq. (3.88) is then:

$$\begin{aligned} \int_{C_S} dz\ z^n &= \int_{-1}^{+1} dt\ t^n = (1)^{n+1} - (-1)^{n+1} \\ &= \begin{cases} 0, & \text{if } 0 < n = 2k+1\,,\ \text{odd} \\ \frac{2}{n+1}\,, & \text{if } 0 \leq n = 2k,\ \text{even}\,. \end{cases} \end{aligned} \tag{3.89}$$

as long as $n \geq 0$. If n is negative, then the singularity on the integration path at $z = 0$ prevents the integral converging. If this was a function of real variables, then there would be no way to obtain a result for the integral when $n < 0$. However, as will now be demonstrated, using complex variables provides a way to compute a result even for $n < 0$. This demonstrates the allowance for multiple contours, of crucial importance to the TayInt program.

Parabolic Arc

2. Taking the parabolic path C_P, shown in blue in Fig. 3.3 and parametrised as:

$$z(t) = t + i(t^2 - 1)\,,\ -1 \leq t \leq 1\,, \tag{3.90}$$

which connects the same two end points as before, -1 and 1, the integral in Eq. (3.88) reads:

$$\int_{C_P} dz\ z^n = \int_{-1}^{+1} \frac{dz}{dt} dt\ z^n = \int_{-1}^{+1} dt\ (t + i(t^2-1)) \cdot (1 + 2it)\,, \tag{3.91}$$

an exact derivative,

$$\frac{d}{dt}\left(\frac{[t + i(t^2-1)]^{n+1}}{n+1}\right) = [t + i(t^2-1)]^n (1 + 2it)\,, \tag{3.92}$$

as long as $n \neq -1$ and so:

$$\begin{aligned}\int_{C_P} dz\ z^n &= \left[\frac{[t + i(t^2-1)]^{n+1}}{n+1}\right]_{-1}^{+1} = \frac{(1)^{n+1} - (-1)^{n+1}}{n+1} \\ &= \begin{cases} 0, & \text{if } -1 \neq n = 2k+1 \text{ odd} \\ \frac{2}{n+1}\,, & \text{if } n = 2k \text{ even}\,. \end{cases}\end{aligned} \tag{3.93}$$

When $n \geq 0$, the result obtained using the parabola is the same as the result obtained using the straight line in Eq. (3.89). This demonstrates the independence of the result from the parametrisation. However, because the parabola avoids the singularity at the origin, the parabolic path allows a result to be obtained when n is negative. This demonstrates the importance of choosing a complex contour to avoid singularities. However, obtaining a result when $n = -1$ using the parabolic path is highly intricate and it is more efficient to use the power of complex integration to find a path along which the result at this point can be found more easily found.

Semi-circle

3. To achieve this, a semi-circular path in the upper half-plane, C_A^+ is considered. C_A^+ is parametrised as $z(t) = e^{it}$, $0 \leq t \leq \pi$ and shown in magenta in Fig. 3.3. Using this path, the integral in Eq. (3.88) is:

$$\begin{aligned}\int_{C_A^+} dz\ z^n &= \int_0^\pi \frac{dz}{dt} dt\ z^n = \int_0^\pi dt\ e^{int} \cdot ie^{it} = \int_0^\pi dt\ ie^{i(n+1)t} \\ &= \left[\frac{ie^{i(n+1)t}}{i(n+1)}\right]_0^\pi = \frac{(-1)^{n+1} - (1)^{n+1}}{n+1} \\ &= \begin{cases} 0\,, & \text{if } -1 \neq n = 2k+1 \text{ odd} \\ -\frac{2}{n+1}\,, & \text{if } n = 2k \text{ even}\,. \end{cases}\end{aligned} \tag{3.94}$$

This result has the opposite sign to that obtained using the straight line and the parabola. This is because the integrals along the straight line and the parabolic arc were performed from $z = -1$ to $z = 1$. However, for the semi-circle, the path starts at $z = e^0 = 1$ and finishes at $z = e^{i\pi} = -1$. Thus, performing the

integral in Eq. (3.94) with e^{it} replaced by e^{-it}, would yield the same result as was obtained using a straight line and a parabola. This demonstrates the reversal of the integration path giving rise to a change in sign, an important result for the development of the TayInt program.

A semi-circle in the lower half plane, C_A^-, could also be used to integrate from -1 to 1. As long as $n \neq -1$, taking the lower semi-circle will not change the result. However, if $n = -1$, the integration along C_A^+ with $z(t) = e^{it}$, $0 \leq t \leq \pi$, from 1 to -1 yields:

$$\int_{C_A^+} \frac{dz}{z} = \int_0^{\pi} dt\ e^{-it} \cdot ie^{it} = \int_0^{\pi} dt\ i = i\pi\,, \tag{3.95}$$

whereas taking the path parametrised by C_A^- from 1 to -1 leads to:

$$\int_{C_A^-} \frac{dz}{z} = \int_0^{-\pi} dt\ e^{-it} \cdot ie^{it} = \int_0^{-\pi} dt\ i = -i\pi\ . \tag{3.96}$$

Thus, changing the direction of integration will change its result if the two paths enclose a singularity between them.

Entire Circle

Having integrated $f(z)$ using an upper and lower semi-circle, the next natural path is an entire circle centred on the origin, C_E. Note that this is a closed path, which is termed a *contour* in complex integration. The corresponding integral is denoted as $\oint$. This is parametrised by $z(t) = re^{it}$ for $0 \leq t \leq 2\pi$. The integral is then carried out as before:

$$\begin{aligned} \oint_{C_E} dz\ z^n &= \int_0^{2\pi} dt\ r^n e^{int} \cdot (rie^{it}) = \int_0^{2\pi} dt\ ir^{n+1} e^{i(n+1)t} \\ &= \left[\frac{r^{n+1}}{n+1} e^{i(n+1)t}\right]_0^{2\pi} = \frac{r^{n+1}}{n+1}\,[1-1] = 0,\ \text{for } n \neq -1\ . \end{aligned} \tag{3.97}$$

Around the pole, $n = -1$:

$$\oint_{C_E} \frac{dz}{z} = \int_0^{2\pi} dt\ e^{-it} \cdot ie^{it} = \int_0^{2\pi} dt\ i = 2i\pi\ . \tag{3.98}$$

This result is important as it reveals that:

$$\int_{C_A^+} \frac{dz}{z} - \int_{C_A^-} \frac{dz}{z} = \pi i - -\pi i = 2\pi i \oint_{C_E} \frac{dz}{z}\,, \tag{3.99}$$

using Eqs. (3.95), (3.96) and (3.98). This illustrates the important property that in general a complex integral can be decomposed into non-overlapping pieces with a common orientation, [88];

$$\int_{C_1 \cup C_2} dz\ f(z) = \int_{C_1} dz\ f(z) + \int_{C_2} dz\ f(z)\,, \tag{3.100}$$

Thus, the general result for the integral in Eq. (3.88) reads:

$$\oint_{C_E} dz\ z^n = \begin{cases} 0\,, & \text{if } n \neq -1 \\ 2\pi i\,, & n = -1\,, \end{cases} \tag{3.101}$$

3.7.3 Relevance for TayInt

This section has illustrated the important features of complex integration necessary for the TayInt program, as follows

1. Results for integrals in singular regions can be obtained by using the freedom to change the contour of integration, which will be used for avoiding the threshold singularities of two-loop Feynman integrals.
2. Performing a complex-mapped two-loop Feynman integral with e^{it} instead of e^{-it}, changes the sign of the result. This is so important because when performing integrals of singular integrands in multiple variables, in order to avoid the singularities it is necessary to use integration paths of different directions for the different variables, thus to obtain the correct results requires carefully keeping track of the signs.
3. When two paths move in different directions around a singular point this leads to a different result for a complex-mapped two-loop Feynman integral. This is another crucial observation because when computing Feynman integrals in the physical kinematic region, the paths of integration often go around singular points. Thus, it is essential to be consistent in this motion in order to obtain a sensible result.

3.7.4 Cauchy's Theorem

Motivation

In the previous section, the result for a complex integral was built up using different integration paths. However, this is in fact a manifestation of the features of a general theorem stated in full generality be Cauchy. As the work in this thesis aims to create a computer program which can be used to produce algebraic approximations for two-loop Feynman integrals in general, building up a result for integrals contour-by-contour is not practical. Thus, the necessary mathematics must also be generally understood.

Statement

To state Cauchy's theorem in its general form requires understanding that if a complex function $f(z)$ is analytic at all points within $\Xi \subset \mathbb{C}$ and is continuous at its boundary, then it is *analytic on the domain*, Ξ. The convention for integrating domains made up

of closed curves is that the integration along the outermost boundary curve, denoted by $\partial\Xi$, is performed anti-clockwise. Also, the integration along interior cavities in the clockwise direction. With this understanding, Cauchy's theorem is stated as:

Theorem 3.7.1 *If $f(z)$ is analytic on a bounded domain $\Xi \subset \mathbb{C}$, then*

$$\oint_{\partial\Xi} dz\ f(z) = 0\,,$$

which is proven using the Cauchy-Riemann equations, (3.13), (3.14), see [88]. If the domain Ξ (on which $f(z)$ is analytic) is simply connected (continuously deformable within the domain while preserving end points), then any closed curve $C \subset \Xi$ will always be within Ξ. So, no matter which path is chosen for a complex integral, Cauchy's theorem applies, which implies that the complex integral within Ξ is path independent.

Corollary 3.7.2 *If $f(z)$ is analytic on a simply connected domain $\Xi \subset \mathbb{C}$, then its complex integral $\int_C f(z)dz$ is path independent for $C \subset \Xi$. Particularly,*

$$\oint_C dz\ f(z) = 0\ .$$

Example

The fact that the domain must be simply connected is very important, as the calculation of $\oint dz\ 1/z = 2\pi i$ around the unit circle shows (Eqs. (3.98), (3.99), (3.7.2)). In this case, The pole prevents curves being continuously deformed into each other through the point $z = 0$, hence the domain $\Xi = \mathbb{C} \setminus 0$ is not simply connected. Consequently, $\oint_C dz\ f(z) \neq 0$. The converse of corollary 3.7.2 is also true: a continuous, complex-valued function satisfying 3.7.2 for all closed curves is necessarily analytic. This is known as *Morera's Theorem*.

3.7.5 Morera's Theorem

Motivation

In order to take advantage of Cauchy's theorem and understand how it facilitates the use of multiple paths of integration, Morera's theorem must also be formally demonstrated.

Proposition 3.7.3 *If $f(z)$ is analytic in a domain Ξ within which there are two closed paths C_1 and C_2 which have the same orientation:*

$$\oint_{C_1} dz\ f(z) = \oint_{C_2} dz\ f(z)\ .$$

To prove this, both the cases in which C_1 and C_2 do and do not intersect need to be considered.

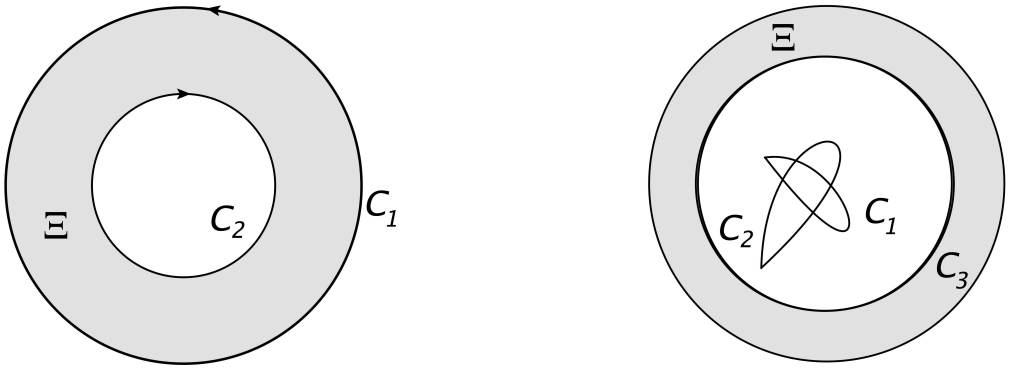


Figure 3.4: Non-intersecting and intersecting closed paths of integration for a complex-valued function.

Proof Part One: C_1 and C_2 do not intersect

1. In this case, let the domain enclosed between C_1 and C_2 be Ξ, so that $\partial\Xi = C_1 \cup C_2$, as shown in the leftmost plot of Fig. 3.4. Using Cauchy's theorem,

$$\oint_{\partial\Xi} dz\ f(z) = 0\ . \tag{3.102}$$

Now, following the convention that outer paths are integrated along anti-clockwise and inner paths are integrated along clockwise, $\oint_{C_1}$ and $-\oint_{C_2}$ have the same orientation. Thus the result in Eq. (3.100) can be used to yield:

$$\oint_{\partial\Xi} dz\ f(z) = \oint_{C_1} dz\ f(z) - \oint_{C_2} dz\ f(z)\ . \tag{3.103}$$

Equations (3.102) and (3.103) together prove Proposition (3.7.3) in the non-intersecting case.

Proof Part Two: C_1 and C_2 do intersect

2. In the case of the two integration paths C_1 and C_2 intersecting, consider a third integration path $C_3 \subset \Xi$ which neither intersect. Because $C_3 \subset \Xi$, Cauchy's theorem applies, so:

$$\oint_{C_3} dz\ f(z) = 0\ . \tag{3.104}$$

However, applying the same logic as in Eq. (3.103) to the closed path formed by $\{C_1, C_3\}$ and $\{C_2, C_3\}$, then:

$$0 = \oint_{C_3} dz\ f(z) - \oint_{C_1} dz\ f(z)\,, \tag{3.105}$$

$$0 = \oint_{C_3} dz\ f(z) - \oint_{C_2} dz\ f(z)\,, \tag{3.106}$$

and so:

$$\oint_{C_3} dz\ f(z) = \oint_{C_1} dz\ f(z) = \oint_{C_2} dz\ f(z)\,, \tag{3.107}$$

proving Proposition 3.7.3 in the intersecting case.

Relevance for TayInt

Proving Proposition 3.7.3 is so important because it opens up many possible paths for any given integral, as long as changing between them does not cross any poles. By using complex variables, there are multiple different paths along which an integral can be performed, allowing unpleasant mathematical features of a two-loop Feynman integrand to be avoided. In the multi-variable case, the difficulty of finding a path that constructs a domain within which a two-loop Feynman integrand is analytic is such that an original algorithm was required to perform it, the final TayInt algorithm.

3.7.6 Illustration of Cauchy's Theorem and Morera's Theorem

The result:

$$\oint_C dz\ z^n = \begin{cases} 0\,, & \text{if } n \neq -1 \\ 2\pi i\,, & n = -1\,, \end{cases} \tag{3.108}$$

previously derived in Eq. (3.7.2) with C being a circle with its centre at the origin, can be arrived at much more quickly and in full generality (for any closed curve C) by a direct application of Cauchy's theorem. In the function $f(z) = z^n$, n is an integer so as to ensure no branch points. When $n \geq 0$ Cauchy's theorem implies that $\oint_C dz\ z^n = 0$ for any contour of integration. If $n \leq -2$ then Cauchy's theorem implies that $\oint_C dz\ z^n = 0$ as long as the contour of integration does not cross the pole at $z = 0$. Even if the contour encloses the pole, Proposition 3.7.3 ensures that $\oint_C dz\ z^n = 0$, as another contour can be constructed which does not contain a pole and intersects the original one. This confirms that $\oint_C dz\ z^n = 0$ when $n \neq -1$. Proposition 3.7.3 also shows that in the case $n = -1$, for any curve C traversing the origin once anti-clockwise, $\oint_C dz\ z^n$ must be equal to the result of integrating $f(z)$ around a circle, which, recalling (3.98), is $2\pi i$. Thus the result for $\oint_C dz\ z^n$ holds in general for any contour C. This will be used to perform complex integrals in a general manner by the TayInt program.

3.8 Summary

The necessary mathematics for understanding two-loop Feynman integrals has now been presented. The mathematical tools that will be needed for the TayInt program have been formalised and demonstrated. In the next chapter, the origin of Feynman integrals within theoretical particle physics will be explained.

4 | From Loop Integrals to Feynman Integrals

4.1 Introduction

As discussed in the Chapter 2, theoretical predictions for differential cross sections in particle physics consist of a perturbative expansion in the relevant coupling constant, with higher orders conferring higher mathematical precision. Adding higher-order predictions necessitates the inclusion of loop diagrams and hence the computation of complicated integrals over the loop momenta. The calculation of such loop integrals is an in-depth subject but can be reduced to the problem of calculating master Feynman integrals, which will be fully discussed in what follows. To begin with, the structure of a loop integral must be defined.

4.2 Defining Loop Integrals

4.2.1 Theory

An example of a loop integral was presented in Eq.(2.46). Such integrals are characterised by the following general features:

Integration Measure

A measure of integration for each internal momentum k_α that must be integrated over. For a diagram with L loops, considered in D dimensions, this reads

$$G_M = \prod_{\alpha=1}^{L} \int \frac{\mu^{4-D}}{i\pi^{\frac{D}{2}}} \mathrm{d}^D k_\alpha \ . \tag{4.1}$$

The factor of $i\pi^{\frac{D}{2}}$ in Eq. ((4.1)) is chosen by convention. It cancels the factor of i that arises from the Wick rotation and the factor of $\pi^{\frac{D}{2}}$ coming from the D-dimensional angular integration. This will be demonstrated later in the section on calculating loop integrals. The renormalisation scale is denoted by μ, which preserves the dimensionless nature of the coupling constant. This is set to unity from here onward.

Propagator Denominator

A denominator that consists of products of $\tilde{j}$ *propagators*, $P_{\tilde{j}}$, raised to arbitrary powers $\nu_{\tilde{j}}$:

$$G_D = \prod_{\tilde{j}=1}^{N} P_{\tilde{j}}^{\nu_{\tilde{j}}}(\{k\},\{p\},m_{\tilde{j}}^2) \ . \tag{4.2}$$

These are linear combinations of the external momenta $p_{\tilde{j}}$, loop momenta k_α and masses $m_{\tilde{j}}$, such that:

$$P_{\tilde{j}}(\{k\},\{p\},m_{\tilde{j}}^2) = q_j^2 - m_j^2 + i\delta \, , \tag{4.3}$$

where the $+i\delta$ in Eq. ((4.3)) results from the solutions of the field equations in terms of causal Green functions and shifts the poles away from the real axis.

Numerator

A numerator which is a tensor of rank R in the loop momenta $k_{\alpha^i}^{\mu_i}$ where $i = 1...R$,

$$G_N = k_{\alpha^1}^{\mu_1} \cdots k_{\alpha^R}^{\mu_R}. \tag{4.4}$$

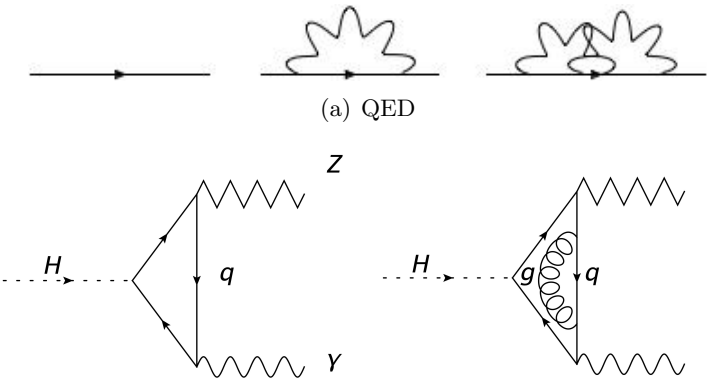
Full Definition

Assembling each piece: G_M, G_D, G_N; a generic Feynman loop integral G in an arbitrary number of dimensions D at loop level L with N propagators, has the momentum space representation:

$$G_{\alpha_1\ldots\alpha_R}^{\mu_1\cdots\mu_R}(\{p\},\{m\}) = \left(\prod_{\alpha=1}^{L}\int \mathrm{d}^D\kappa_\alpha\right) \frac{k_{\alpha_1}^{\mu_1}\cdots k_{\alpha_R}^{\mu_R}}{\prod_{\tilde{j}=1}^{N} P_{\tilde{j}}^{\nu_{\tilde{j}}}(\{k\},\{p\},m_{\tilde{j}}^2)} \tag{4.5}$$

$$\mathrm{d}^D\kappa_\alpha = \frac{\mu^{4-D}}{i\pi^{\frac{D}{2}}}\,\mathrm{d}^Dk_\alpha \ , \ P_{\tilde{j}}(\{k\},\{p\},m_{\tilde{j}}^2) = q_{\tilde{j}}^2 - m_{\tilde{j}}^2 + i\delta \ , \tag{4.6}$$

wherein the propagators $P_{\tilde{j}}$ with mass $m_{\tilde{j}}$ can be raised to arbitrary powers $\nu_{\tilde{j}}$, $N_\nu = \sum_{\tilde{j}}^{N} \nu_t ildej$ and the $q_{\tilde{j}}$ are linear combinations of external momenta p_i and loop momenta k_α. The rank R of the integral is indicated by the number of loop momenta appearing in the numerator. The indices α_i denote which of the L loop momenta belongs to which Lorentz index μ_i. In this thesis, $R = 0$ is taken for conciseness, although the TayInt program is valid for arbitrary rank. It was shown in Chapter 2 that calculating the Feynman diagrams needed to make precise theoretical predictions for differential cross sections in particle physics actually reduces to the problem of computing *loop integrals*. It has now been shown that these loop integrals have a general representation, Eq. (4.5), thus can be studied as mathematical objects, in isolation from their physically-motivated



(a) QED

(b) $H \to Z\gamma$

Figure 4.1: Two sets of Feynman diagrams up to the two-loop level that feature in the perturbative expansion of: 4.1(a), the electron propagator in QED, 4.1(b), $H \to Z\gamma$ production [92]. The former is used here for pedagogical purposes to illustrate the process of calculating loop integrals, the latter, which does not exist at tree level, will be used to illustrate the TayInt program, especially its contour choosing algorithm. The solid lines denote fermions (labelled as q in Fig. 4.1(b) to signify that they are quarks not electrons), the wavy lines photons, the zigzag lines the Z-boson, the curly lines gluons and the dashed line the scalar Higgs boson.

origins. Once new loop integrals are calculated, they can be used to compute higher-order corrections to the S-matrix, which at present represents the main obstacle to calculating differential cross sections at the two-loop level.

4.2.2 Example

Now that the structure of loop integral has been understood, two such examples are shown below, with Fig. 4.1(a) representing a part of the perturbative expansion for the electron propagator in QED and Fig. 4.1(b) containing diagrams that enter the $H \to Z\gamma$ decay rate at two-loop. It is clear that the power of the coupling constant present in each diagram rises as the figure is scanned from left to right. This is because each diagram constitutes a higher loop-level and hence degree of precision for the S-matrix and differential cross section. For the 1-loop contribution to the electron propagator, depicted in Fig. 4.1(a), applying the QED Feynman rules [93] gives rise to the integral

$$\begin{aligned}\Sigma(p^2, m) &= \int \frac{d^4k}{(i\pi^2)}(-ie\gamma^\nu)\left[\frac{-ig_{\mu\nu}}{k^2+i\delta}\right]\left[\frac{i(\not{k}+\not{p}+m_F}{(k+p)^2-m_e^2+i\delta}\right](-ie\gamma^\mu) \\ &= \int \frac{d^4k}{(i\pi^2)}\gamma_\mu \frac{1}{k^2+i\delta}\cdot\frac{\not{k}+\not{p}+m_e}{(k+p)^2-m_e^2+i\delta}\gamma^\mu \ . \end{aligned} \tag{4.7}$$

Table 4.1: The features of the one-loop integrals $B_0^{11}(D=4)$ and $B_1^{11}(D=4)$ given in Eqs. (4.7) and (4.8) that make up the correction to the electron propagator in QED.

Integral	**Integration Measure,** G_M	**Propagator Denominator,** G_D	**Numerator,** G_N
$B_0^{11}(D=4)$	$\int \frac{\mathrm{d}^4k}{i\pi^2}$	$(k^2+i\delta)$ $\times((k+p)^2-m_e^2+i\delta)$	1
$B_1^{11}(D=4)$	$\int \frac{\mathrm{d}^4k}{i\pi^2}$	$(k^2+i\delta)$ $\times((k+p)^2-m_e^2+i\delta)$	k^μ

Performing the Dirac algebra [93], this can be rewritten as a superposition of two loop integrals:

$$\Sigma(p^2, m_e) = e^2\left(2\gamma_\mu B_1^{11} + 2(\not{p} - 2m_e)B_0^{11}\right), \tag{4.8}$$

wherein;

$$B_0^{11}(D=4) = \int \frac{d^4k}{(i\pi^2)}\frac{1}{k^2+i\delta}\cdot\frac{1}{(k+p)^2-m_e^2+i\delta}, \tag{4.9}$$

$$B_1^{11}(D=4) = \int \frac{d^4k\ k^\mu}{(i\pi^2)}\frac{1}{k^2+i\delta}\cdot\frac{1}{(k+p)^2-m_e^2+i\delta}, \tag{4.10}$$

with $D=4$ denoting the number of space-time dimensions. The loop integral in Fig. 4.1(a) can be split into the features defined in the previous section, as depicted in Table 4.1.

4.3 Calculating Loop Integrals

The mathematical study of loop integrals is an immensely detailed subject, as each of their pieces, G_M, G_D and G_N present barriers to their calculation. Consequently, each such pieces must be manipulated considerably before integration is even possible. For more detail on the subject, see for example [94]. The mathematical foundation needed for the calculation of loop integrals is outlined below. For the rest of this thesis, the $+i\delta$ terms which shift the poles away from the real axis will be suppressed for brevity, unless explicitly stated otherwise.

4.3.1 Theory

Dimensional Regularisation

In general, loop integrals are divergent. As they are part of a calculation of physically measurable quantities, this clearly presents a problem. But, the first step in addressing it is to find a method for performing the integration, termed *regularisation*. The second

step is to then make that result finite, termed *renormalisation*. Referring to Eqs.(4.5) and (4.6), there are two pathways to infinities. The first is when one of the $P_{\tilde{j}}(\{k\},\{p\},m_j^2) = q_{\tilde{j}}^2 - m_{\tilde{j}}^2 + i\delta \ = 0$, known as an *infrared* divergence. The second is seen when $k_\alpha \to \infty$, in which limit $P_{\tilde{j}}(\{k\},\{p\},m_{\tilde{j}}^2) = q_{\tilde{j}}^2 - m_{\tilde{j}}^2 + i\delta \ \to P_{\tilde{j}}(\{k\}) = k_{\tilde{j}}^2$. The dimensionality of the integrand in Eq.(4.5) is then given by

$$S_k = 4 + R - \sum_{\tilde{j}=1}^{N} \nu_{\tilde{j}}\,, \tag{4.11}$$

in $D = 4$ dimensions. If $S_k \geq 0$ then the integral contains *ultraviolet* divergences. In order to render loop integrals analytic and hence calculable despite these divergences, a procedure was developed by t'Hooft and Veltman [95], Bollini and Giambiagai, [96], York, [97] and Ashmore, [98]. This generalises the number of dimensions from $D = 4$ to $D = d$, where d is an arbitrary complex number. The loop integral can then be calculated in d-dimensions and analytically continued to $d = 4 - 2\epsilon$ at the end of the calculation. After which, the result can be Laurent expanded in ϵ to separate the divergent and finite parts. The parameter d is known as the regularising parameter. The principles of analytic continuation and Laurent expansions are discussed fully in [88].

At the end of this step, a scheme has been introduced within which loop integrals are calculable.

Passarino-Veltman Reduction

Now that such a scheme within which loop integrals can be calculated has been introduced, the next step is to simplify their numerators by reducing tensor integrals to a superposition of scalar loop integrals. At the one-loop level, all loop integrals can be reduced to a set of scalar integrals [99], by a process known as *Passarino-Veltman* reduction. This process is derived from the observation that, at one-loop, every scalar product of loop and external momenta can be equated to a linear superposition of propagators. At the end of this step, only scalar loop integrals contribute to a given higher-order correction.

Feynman Parameters

Once the loop integral computation has been reduced to a calculation of scalar loop integrals, the next problem to be considered is that integrals of the form given in Eq.(4.5) are problematic to compute because of the multiple momentum dependent factors in the denominator. In order to calculate such integrals *Feynman parameters* $t_{\tilde{j}}$ are used to

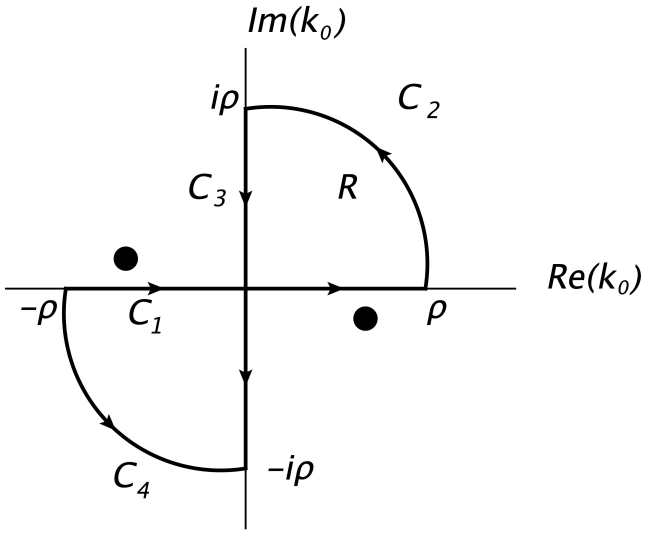


Figure 4.2: The contour of integration (bold) of arc length ρ used to translate Minkowski into Euclidean space for loop integrals. The dark circles represent the poles, shifted off the real axis by the $+i\delta$ prescription.

compress the propagators in the denominator into one quadratic, through the identity

$$\frac{1}{\prod_{\tilde{j}=1}^{N} P_{\tilde{j}}^{\nu_{\tilde{j}}}(\{k\},\{p\},m_{\tilde{j}}^2)} = \frac{\Gamma(\nu)}{\prod\limits_{\tilde{j}=1}^{N}\Gamma(\nu_{\tilde{j}})} \int_0^1 \prod_{\tilde{j}=1}^{N} dt_{\tilde{j}} t_{\tilde{j}}^{\nu_{\tilde{j}}-1} \cdot \frac{\delta\left(\sum\limits_{\tilde{j}=1}^{n} t_{\tilde{j}} \ - 1\right)}{\left(\sum\limits_{\tilde{j}=1}^{N} t_{\tilde{j}} P_{\tilde{j}}\right)^{\nu}} \ , \tag{4.12}$$

$$\nu = \sum_{\tilde{j}=1}^{N} \nu_{\tilde{j}} \ . \tag{4.13}$$

The application of this identity allows the denominator of the scalar loop integrals under consideration to be rearranged into a form facilitating the integration of the loop momenta at the end of this step.

Wick Rotation and Loop Momentum Integration

Now that the denominator of a loop integral is in a form that is ready for integration, it must be analysed more closely to determine the relevant coordinates required to do so. In a spherically symmetric space, spherical coordinates in d dimensions:

$$\int_{-\infty}^{+\infty} d^d k = \int_0^{\infty} dr \ r^{d-1} \int d\Omega_{d-1} \, , \tag{4.14}$$

can be used to perform the loop momentum integration following Feynman parametrisation. Note that:

$$\int d\Omega_{d-1} = V(d) = \frac{2\pi^{d/2}}{\Gamma(d/2)}\,, \tag{4.15}$$

the volume of a d-dimensional unit sphere [93]. It is this which motivates the factor $\pi^{D/2}$ in the denominator of the integration measure in (4.6). However, loop integrals are *not* spherically symmetric as they are defined in *Minkowski* space, so that $r = k_0^2 - k^i k_i$ and in order to use spherical coordinates it must be the case that:

$$r = \left(\sum_{\tilde{j}=1}^{d} k_{\tilde{j}}^2\right)^{1/2} \quad . \tag{4.16}$$

For spherical coordinates to be applicable, loop integrals must therefore be translated into Euclidean space. This can be done by an application of Cauchy's theorem, as was discussed in Chapter 3. The integral $\int_{\partial\Xi} dk_0 f(k_0^2)$, where $f(k_0^2) = (k_0^2 + k_i^2 - \Delta^2)^{-2}$ and $\partial\Xi$ is the contour in bold depicted in Fig. 4.2, can be split into four parts, C_1, C_2, C_3, C_4. Therein: along C_1, $k_0 = \alpha$, $\alpha \in [-\rho, \rho]$ along C_2 $k_0 = \rho e^{i\vartheta}$, $\vartheta \in [0, \pi/2]$, along C_3 $k_0 = \alpha e^{i\pi/}$, $\alpha \in [\rho, -\rho]$ and along C_4, $k_0 = -\rho e^{i\vartheta}$, $\vartheta \in [-\pi/2, 0]$.

However, the quantity that appears in loop integrals is $\int_{-\infty}^{+\infty} dk_0 f(k_0^2)$, thus the limit $\rho \to \infty$ must be applied, which, together with Cauchy's theorem, leads to the sum of the integrals along the paths C_2 and C_4 vanishing, from which the way to translate Minkowski into Euclidean space is revealed:

$$\int_{-\infty}^{+\infty} dk_0 f(k_0^2) = i \int_{-\infty}^{+\infty} dk_0 f(-k_0^2)\ . \tag{4.17}$$

This can be implemented by the substitution $k_0 \to i k_0^E$ in loop integrals, where the subscript E denotes Euclidean space. This substitution is known as a *Wick rotation*. At the end of this step, the loop integral to be calculated has been translated into Euclidean space. This allows the use of spherical coordinates to parametrise the integration of the loop momenta.

Now that the loop integral to be calculated is spherically symmetric, it can be written in terms of d-dimensional spherical coordinates, using Eq.(4.15) to perform the angular integration. The radial integration can then be carried out by means of a change of variable and the use of the Euler Beta function,

$$B(a,b) = \int_0^\infty dz\ \frac{z^{a-1}}{(1+z)^{a+b}} = \int_0^1 dy\ y^{a-1}(1-y)^{b-1} = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}. \tag{4.18}$$

At the end of this step, the integration of the loop momentum has been completed, leaving only the Feynman parameter integration.

Feynman Parameter Integration

All of the above steps provide a recipe for transforming a higher-order correction to a differential cross section in particle physics into the computation of Feynman integrals. At the one-loop level the calculation of Feynman integrals has been automated using the method of differential equations leading to results expressible in terms of generalised polylogarithms. However at the two-loop level the calculation of Feynman integrals is still not fully understood in general, a problem which the work in this thesis aims to tackle.

4.3.2 Example

Now that the steps for calculating a loop integral have been outlined, they will be next be illustrated. For pedagogical reasons, they are applied to the example of the loop integrals that contribute to the electron propagator at one-loop. These are given in Eqs.(4.9) and (4.10).

Dimensional Regularisation

Before any calculation can be performed, the loop integrals must first be defined in such a way that it is even possible to do so. As $k \to \infty$, in four dimensions the integral B_0^{11}, Eq. (4.9), *diverges*, thus is not an analytic function. But, in three dimensions it is finite as $k \to \infty$. Hence, the change in the number of dimensions transforms a divergent integral, which cannot be calculated, into a convergent one, which is calculable. To make formal use of this observation, the number of dimensions is replaced by an arbitrary complex number d, in accordance with the principle of dimensional regularisation introduced in the previous subsection. Then, the integral $B_0^{11}(d)$ can then be defined such that it is an analytic function of d, which can in principle be explicitly evaluated.

Once the calculation of the integral is over, the result for $D = 4$ *must* be reproduced, as loop integrals are defined in four dimensions. The principle of *analytic continuation* can then be used to return to four-dimensional space.

The dimensionally regularised version of (4.7) reads:

$$\begin{aligned}\Sigma(p^2, m_e) &= \int \frac{d^d k}{i\pi^{d/2}} (-ie\gamma^\nu) \left[\frac{-ig_{\mu\nu}}{k^2}\right] \left[\frac{i(\not{k} + \not{p} + m_F}{(k+p)^2 - m_e^2}\right] (-ie\gamma^\mu) \\ &= \int \frac{d^d k}{i\pi^{d/2}} \gamma_\mu \frac{1}{k^2} \cdot \frac{\not{k} + \not{p} + m_e}{(k+p)^2 - m_e^2} \gamma^\mu \ . \end{aligned} \tag{4.19}$$

The necessary Dirac algebra now reads:

$$\gamma_\mu \gamma^\nu (k_\nu + p_\nu) \gamma^\mu = \gamma_\mu (-\gamma^\mu \gamma^\nu + 2g^{\mu\nu}) = (2-d)\gamma^\nu \ , \tag{4.20}$$

where $g^{\mu\nu}$ is the space-time metric in d dimensions and $\gamma_\mu \gamma^\mu = d$, [93]. At this point another question naturally arises, namely, how to continue the γ_5 matrix from four to d dimensions. There is not a unique solution but rather three different schemes. These

schemes and the way to transition between them at one-loop, are discussed in detail in [100, 101]. After performing the Dirac algebra, $\Sigma(p^2, m_e)$ reads:

$$\Sigma(p^2, m_e) = e^2 \left((d-2)\gamma_\mu B_1^{11} + \left[(d-2)\not{p} - d m_e \right] B_0^{11} \right) , \tag{4.21}$$

where, in dimensionally regularised form,

$$B_0^{11}(D=d) = \int \frac{d^d k}{(i\pi^{d/2})} \frac{1}{k^2} \cdot \frac{1}{(k+p)^2 - m_e^2} , \tag{4.22}$$

$$B_1^{11}(D=d) = \int \frac{d^d k \; k^\mu}{(i\pi^{d/2})} \frac{1}{k^2} \cdot \frac{1}{(k+p)^2 - m_e^2} . \tag{4.23}$$

At the end of this step, the loop integrals B_0^{11} and B_1^{11} have been written in a form such that their calculation is possible.

Passarino-Veltman Reduction

In order to simplify this calculation, the tensor structures in the numerators of loop integrals can be removed by reducing each tensor loop integral to a superposition of scalar loop integrals. For example, consider the integral $B_1^{11}(D=d)$ given in dimensionally regularised form in Eq.(4.23). The result for this integral must transform in the same way as the integral itself, however must not contain any loop momenta, as these are integration variables only. Therefore, the only available tensor structure is p^μ and so the result must be Ap^μ. A contraction with p_μ is used to determine A, as follows:

$$\begin{aligned} Ap^\mu &= \int \frac{d^d k \; k^\mu}{i\pi^{d/2}} \frac{1}{k^2} \cdot \frac{1}{(k+p)^2 - m_e^2} \\ \implies A &= \frac{1}{2p^2} \left[\int \frac{d^d k}{i\pi^{d/2}} \frac{2k \cdot p}{k^2 \cdot (k+p)^2 - m_e^2} \right] . \end{aligned} \tag{4.24}$$

The numerator of the integrand in Eq.(4.23) can then be rewritten as a combination of propagators, leading to:

$$A = \frac{1}{2p^2} \left[\int \frac{d^d k}{i\pi^{d/2}} \frac{((k+p)^2 - m_e^2) - k^2 - p^2 + m_e^2}{k^2 \cdot ((k+p)^2 - m_e^2)} \right] \tag{4.25}$$

$$\begin{aligned} &= \frac{1}{2p^2} \left\{ \left[\int \frac{d^d k}{i\pi^{d/2}} \frac{1}{k^2} \right] - \left[\int \frac{d^d k}{i\pi^{d/2}} \frac{1}{(k+p)^2 - m_e^2} \right] \right. \\ &\quad \left. + \left[\int \frac{d^d k}{i\pi^{d/2}} \frac{m_e^2 - p^2}{k^2((k+p)^2 - m_e^2)} \right] \right\} \end{aligned} \tag{4.26}$$

$$\begin{aligned} \implies B_1^{11} &= -\frac{p^\mu}{2p^2} \left\{ \int \frac{d^d k}{i\pi^{d/2}} \frac{1}{(k+p)^2 - m_e^2} \right\} \\ &\quad + \frac{p^\mu}{2p^2} \left\{ \int \frac{d^d k}{i\pi^{d/2}} \frac{m_e^2 - p^2}{k^2((k+p)^2 - m_e^2)} \right\} . \end{aligned} \tag{4.27}$$

The first integral in Eq. (4.26) vanishes because it does not contain any external scales. This leads to a result for the rank-one integral B_1^{11} as a superposition of scalar integrals, Eq. (4.27). Inserting this into Eq. (4.21) yields:

$$\begin{aligned}\Sigma(p^2, m_e) = &\frac{\not{p}(d-2)}{2p^2}\left\{\int \frac{d^dk}{i\pi^{d/2}}\frac{1}{(k+p)^2-m_e^2}\right\} \\ &+\left(\frac{\not{p}(d-2)}{2p^2}+dm_e\right)\left\{\int \frac{d^dk}{i\pi^{d/2}}\frac{m_e^2-p^2}{k^2((k+p)^2-m_e^2}\right\}\,. \end{aligned} \tag{4.28}$$

In fact, to calculate $\Sigma(p^2, m_e)$ it is not necessary to use Passarino-Veltman reduction, as will be demonstrated in the forthcoming steps, however it is useful to illustrate the method. At the end of this step the only loop integrals contributing to the relevant higher-order correction are scalar ones.

Feynman Parameters

Now that the numerators of the loop integrals have been simplified, the denominator must be simplified next. Applying Eq.(4.12) to the denominator of Eqs.(4.9) and (4.10):

$$\begin{aligned}\frac{1}{k^2\cdot((k+p)^2-m_e^2)} &= \int_0^1 dt_1dt_2\frac{\delta(1-t_1-t_2)}{k^2t_2+((k+p)^2-m_e^2)t_1)^2} \\ &= \int_0^1 dt_1((1-t_1)k^2+t_1((k+p)^2-m_e^2))^{-2} \\ &= \int_0^1 dt_1(k^2+2kt_1p-t_1p^2-m_e^2t_1)^{-2} \\ &\overset{(*)}{=} \int_0^1 \frac{dt_1}{(k^2-\Delta)^2}\,, \end{aligned} \tag{4.29}$$

where in the last equation, (*), the translation $k \to k-p$, which leaves the boundaries and measure of k-integration unchanged, has been performed prior to completing the square. Also, $\Delta = t_1(t_1p^2+p^2+m_e^2)$, which more elegantly reads $\Delta = t_1(1-t_1)(p^2+m_e^2)+m_e^2t_1^2$. Thus, after Feynman parametrisation, the integrals in Eqs.(4.9) and (4.10) become

$$B_0^{11}(d) = \int_0^1 dt_1 \int \frac{d^dk}{i\pi^{d/2}}\frac{1}{(k^2-\Delta)^2}\,, \tag{4.30}$$

$$B_1^{11}(d) = \int_0^1 dt_1 \int \frac{d^dk\ k^\mu}{i\pi^{d/2}}\frac{1}{(k^2-\Delta)^2} = 0\,, \tag{4.31}$$

where, crucially, the order of integration has been changed so that the k integral is performed first. The integral $B_1^{11}(d) = 0$ because it is odd under $k \to -k$ and thus vanishes. This is why, in this case, Passarino-Veltman reduction is not required. It is always good practice to work generally for a given type of loop integral, such that the powers of propagators are variables, as this allows the instant evaluation of loop integrals

of similar type should the result be needed later. Thus, in the next steps, the generalised version of B_0^{11}:

$$B_0^{\frac{\nu}{2}\frac{\nu}{2}}(d) = \int_0^1 dt_1 \int \frac{d^dk}{i\pi^{d/2}} \frac{1}{(k^2-\Delta)^\nu}\,, \tag{4.32}$$

will be considered. At the end of this step the loop integrals contributing to the one-loop electron propagator have been reduced to a general integral in the loop momenta and Feynman parameters.

Wick Rotation and Loop Momentum Integration

In order to integrate the loop momentum in this generalised integral, it must be written in a spherically symmetric space so that it can be parametrised by spherical coordinates. This is done by means of a Wick rotation, Eq.(4.17).

Wick rotating the loop integral in Eq.(4.32) leads to:

$$B_0^{\frac{\nu}{2}\frac{\nu}{2}}(d) = \int_0^1 dt_1 \int \frac{id^dk_E}{i\pi^{d/2}} \frac{(-1+i\delta)^{-\nu}}{(k_E^2+\Delta)^\nu} = \int_0^1 dt_1 \int \frac{d^dk_E}{\pi^{d/2}} \frac{(-1+i\delta)^{-\nu}}{(k_E^2+\Delta)^\nu}\,, \tag{4.33}$$

revealing why the factor i^{-1} is included in the measure of integration. The denominator of the loop integral is now positive definite, with the only remnants of the Minkowski inner product now contained in the $(-1+i\delta)^{-\nu}$ term. To make this explicit, the $+i\delta$ is included in Eq. (4.33). Having made the Wick rotation, the integral in Eq.(4.33) is now spherically symmetric, with $r = \sqrt{k_E^2}$, so it can be written in terms of d-dimensional spherical coordinates using Eq.(4.15):

$$\begin{aligned}
B_0^{\frac{\nu}{2}\frac{\nu}{2}}(d) &= \int_0^1 dt_1 \frac{2(-1)^{-\nu}}{\Gamma(d/2)} \int_0^\infty d^dk_E \frac{k_E^{d-1}}{(k_E^2+\Delta)^\nu} \\
&\overset{(*)}{=} \int_0^1 dt_1 \frac{2(-1)^{-\nu}}{\Gamma(d/2)} \int_0^\infty \frac{du}{2}\; u^{d/2-1}\cdot(u+\Delta)^{-\nu} \\
&\overset{(**)}{=} \int_0^1 dt_1 \frac{(-1)^{-\nu}\Delta^{d/2-\nu}}{\Gamma(d/2)} \int_0^\infty dr\; r^{d/2-1}\cdot(1+r)^{-\nu} \\
&\overset{(***)}{=} \int_0^1 dt_1 \frac{(-1)^{-\nu}\Delta^{d/2-\nu}}{\Gamma(d/2)} \cdot \frac{\Gamma(d/2)\Gamma(\nu-d/2)}{\Gamma(\nu)}\,,
\end{aligned} \tag{4.34}$$

where in obtaining (*) the substitution $u = k_E^2$ was used. In deriving (**) the substitution $r = u\Delta$ was used and in making the equality (***) the Euler Beta function, Eq.(4.18) was employed. Thus:

$$B_0^{\frac{\nu}{2}\frac{\nu}{2}}(d) = \int_0^1 dt_1 \int \frac{d^dk}{i\pi^{d/2}} \frac{1}{(k^2-\Delta)^\nu} \qquad = \frac{(-1)^{-\nu}\Gamma(\nu-d/2)}{\Gamma(\nu)} \int_0^1 dt_1 \Delta^{d/2-\nu}\,. \tag{4.35}$$

At the end of this step, only the Feynman parameter integration remains before a dimensionally regularised result for the one-loop correction to the electron propagator in QED can be obtained.

Feynman Parameter Integration

All of the prior work has reduced the task of computing a higher-order correction in QED to computing a *Feynman integral*, which, after re-inserting $\Delta = t_1(1-t_1)(p^2+m_e^2)+m_e^2 t_1^2$ and setting $\nu = 2$ in (4.35), reads:

$$B_0^{11}(d) = \Gamma(2-d/2)\int_0^1 dt_1[t_1(1-t_1)(p^2+m_e^2)+m_e^2 t_1^2]^{d/2-2} \ . \tag{4.36}$$

The integrand is then cast into the identifiable form:

$$B_0^{11}(d) = (m_e^2\tilde{p})^{d/2-2}\Gamma(2-d/2)\int_0^1 dt_1 t_1^{d/2-2}\left[1+t_1\left(\frac{1-\tilde{p}}{\tilde{p}}\right)\right]^{d/2-2} , \tag{4.37}$$

allowing the Feynman integral to be computed in terms of *Hypergeometric functions* [102], using:

$$\int_0^1 dx\ \frac{x^{\mu-1}}{(1+\beta x)} = \frac{1}{\mu}{}_2F_1(1,\mu;\mu+1;-\beta) \ . \tag{4.38}$$

Hence, the result for the loop integral $B_0^{11}(D=d)$ reads:

$$B_0^{11}(d) = (p^2+m_e^2)^{\frac{d}{2}-2}\frac{\Gamma\left(2-\frac{d}{2}\right)}{d/2-1}{}_2F_1\left(1,\frac{d}{2}-1;\frac{d}{2};\frac{p^2}{p^2+m_e^2}\right) . \tag{4.39}$$

Inserting this, $B_1^{11}(D=d)=0$ into Eq.(4.28) and taking the limit $d\to 4-2\epsilon$ provides a result for the one-loop correction to the electron propagator where ϵ is small and parametrises the divergences:

$$\begin{aligned}\Sigma(p^2,m_e) = 2e^2(m_e^2+p^2)^{-\epsilon}&\left[\not{p}-\frac{2-\epsilon}{1-\epsilon}m_e\right]\\ &\times\Gamma(\epsilon)\,{}_2F_1\left(1,1-\epsilon;2-\epsilon;\frac{p^2}{p^2+m_e^2}\right) \ .\end{aligned} \tag{4.40}$$

Expanding this around $\epsilon=0$ yields a Laurent series in ϵ. The mathematical difficulty introduced by even the most elementary Feynman integral illustrates the extreme challenge of producing results for these objects at the two-loop, four-point level in more intricate theories such as QCD, a challenge which the work of this thesis aims to address.

4.4 Generalisation to Feynman Integrals

The previous section demonstrates that the core of computing differential cross sections in particle physics is the calculation of *Feynman integrals*. After integrating the loop momenta, the general form of a scalar Feynman-parametrised multi-loop integral in D-dimensions reads:

$$G(\{q\},\{m\}) = \frac{(-1)^{N_\nu}}{\prod_{\tilde{j}=1}^N\Gamma(\nu_{\tilde{j}})}\prod_{\tilde{j}=1}^N\int_0^\infty dt_{\tilde{j}}\ t_{\tilde{j}}^{\nu_{\tilde{j}}-1}\,\delta(1-\sum_{\tilde{j}=1}^N t_{\tilde{j}})\ \frac{\mathcal{U}^{N_\nu-(L+1)D/2}}{\mathcal{F}^{N_\nu-LD/2}(\{q\},\{m\})} \ , \tag{4.41}$$

where, carrying on from Eq. 4.5, N is the number of propagators, $N_\nu = \sum_{\tilde{j}=1}^{N} \nu_{\tilde{j}}$, L the number of loops and $\nu_{\tilde{j}}$ the propagator powers. The functions $\mathcal{U}$ and $\mathcal{F}$ are the first and second Symanzik polynomial, respectively, and are homogeneous in the Feynman parameters.

4.4.1 Theory

As can be seen from Eq.(4.41), in order to construct a general Feynman integral corresponding to an arbitrary diagram, the Symanzik polynomials corresponding to that diagram must be derived.

The Symanzik polynomials are functions of the Feynman parameters and can in general be constructed from the diagram [103], as follows.

Construct the 1-trees $T \in \mathcal{T}_1$ and find $\mathcal{U}$

1. Take an L-loop Feynman diagram and cut L lines in every way possible, generating the specific one-trees, T, belonging to the set of all such one-trees, $\mathcal{T}_1$. The set of cut lines in each one-tree are known as chords, $\mathcal{C}(T)$ and associating a Feynman parameter $t_{\tilde{j}}$ to each cut line allows the Symanzik polynomial $\mathcal{U}$ to be constructed as:

$$\mathcal{U}(t_{\tilde{j}}) = \sum_{T \in \mathcal{T}_1} \left[\prod_{\tilde{j} \in \mathcal{C}(T)} t_{\tilde{j}} \right] . \tag{4.42}$$

Construct the 2-trees, $\hat{T} \in \mathcal{T}_2$ and find $\mathcal{F}$

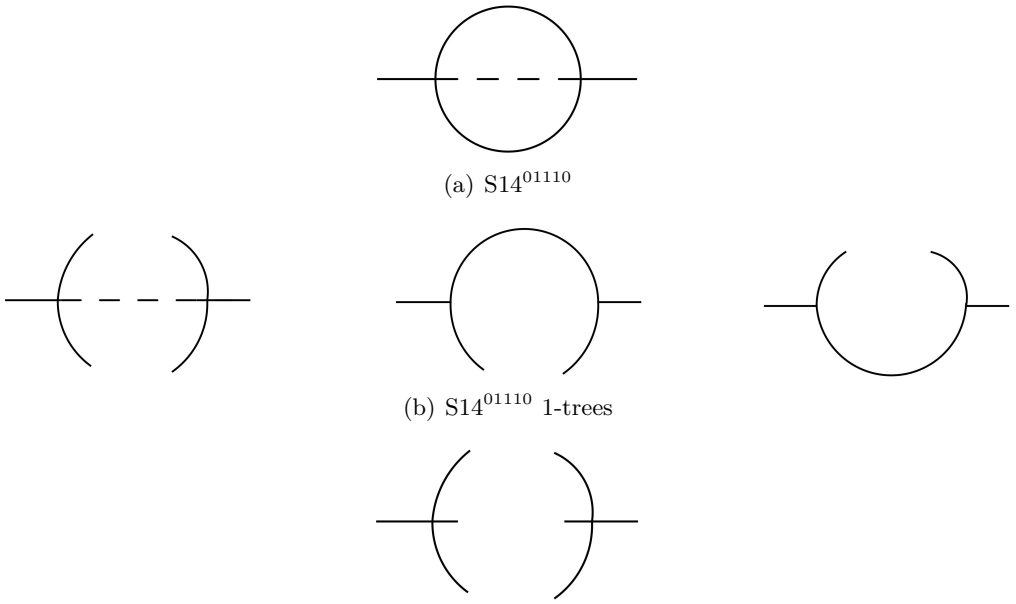
2. Cutting one further line in the trees $T \in \mathcal{T}_1$ generates the set of two-trees, $\mathcal{T}_2$. The corresponding chords generate monomials of degree $L+1$. With the Feynman parameters associated with each line of the original graph as before and defining the momentum flow through the second cut as:

$$s_{\hat{T}} = \left(\sum_{i \in \mathrm{cut}(\hat{T})} p_i \right)^2 , \tag{4.43}$$

the second symanzik polynomial is constructed as follows:

$$\mathcal{F}(t_{\tilde{j}}) = - \sum_{\hat{T} \in \mathcal{T}_2} \left[\prod_{\tilde{j} \in (\hat{T})} t_{\tilde{j}} \right] \cdot s_{\hat{T}} + \mathcal{U}(t_{\tilde{j}}) \sum_{\tilde{j}=1}^{N} x_{\tilde{j}} m_{\tilde{j}}^2 , \tag{4.44}$$

where N is the number of Feynman parameters.



(a) $S14^{01110}$

(b) $S14^{01110}$ 1-trees

(c) $S14^{01110}$ 2-trees

Figure 4.3: The diagram $S14^{01110}$ ((a)), its 1-trees ((b)) and 2-trees ((c)), from which its Symanzik polynomials are constructed topologically. Dashed lines indicate massless and solid internal lines massive propagators.

4.4.2 Example

The construction of the Symanzik polynomials based on the Feynman diagram will now be illustrated in what follows using the example of the two-mass sunrise diagram, which corresponds to the integral family:

```
name:  "S14"
loop_momenta:  [k,l]
propagators:
- [ "k", m ]
- [ "k-l", 0 ]
- [ "k+p", m ]
- [ "l", m ]
- [ "l+p", m ],
```

with the five propagator raised to the powers 0, 1, 1, 1, 0 (this notation will be introduced in general in Chapter 5). Hence it is denoted here as $S14^{01110}$. The two-mass sunrise, $S14^{01110}$, is depicted with its one-trees and two-trees in Fig. 4.3.

Construct the 1-trees $T \in \mathcal{T}_1$ and find $\mathcal{U}$

1. For the sunrise graph, as can be seen by reference to 4.3(a), $L = 2$. The corresponding one-trees are shown in Fig. 4.3(b). Associating a Feynman parameter (from top to bottom in ascending order) to each line of $\mathrm{S14}^{01110}$, the monomials associated with each such one-tree read t_1t_3, t_2t_3, t_1t_2. Therefore, applying (4.42):

$$\mathcal{U}(t_{\tilde{j}}) = t_1t_3 + t_2t_3 + t_1t_2 \ . \tag{4.45}$$

Construct the 2-trees, $\hat{T} \in \mathcal{T}_2$ and find $\mathcal{F}$

2. In the case of $\mathrm{S14}^{01110}$, cutting one further line of the one-trees (Fig. 4.3(b)) generates only one distinct two-tree, shown in Fig. 4.3(c). The monomial associated with this two-tree is $t_1t_2t_3$. Therefore, applying (4.44), the second Symanzik polynomial reads

$$\mathcal{F}(t_{\tilde{j}}) = (t_1t_3 + t_2t_3 + t_1t_2)(t_2 + t_3)m^2 - t_1t_2t_3p^2 \,, \tag{4.46}$$

   as the incoming momentum is p and there are two equally massive propagators.

In this way, the Feynman integral corresponding to a particular topology can be automatically written down by deriving the Symanzik polynomials from the topology and using the generalised formula in Eq.(4.41).

4.5 Working with Feynman Integrals

Once all the Feynman integrals contributing to a process have been constructed, they will in general have many different propagator powers in their denominators. In order to work with these integrals, they must first be organised into distinct *topologies*. These are determined by the propagators the integral contains, regardless of the powers they are raised to. Many of the Feynman integrals with higher propagator powers in a given topology can be reduced to combinations of integrals with lower propagator powers. So, not all of them need to be calculated. The linearly independent Feynman integrals with the minimal sum of propagator powers which contain all the information in the full set of Feynman integrals are known as the *master integrals* of a process. As a $2 \to 2$ QCD process at two-loop will lead to *thousands* of Feynman integrals, performing a reduction to master integrals is of paramount importance. Hence, the only Feynman integrals that need to be calculated in order to compute higher-order corrections to differential cross sections are the master integrals. Thus, when Feynman integrals are referred to in this thesis, it should be understood that this means the master integrals. For a comprehensive review of the subject of master integrals, see [7] and [104].

4.5.1 Theory

A fully systematic approach to finding master integrals was first presented by Laporta, [105]. The Laporta algorithm has since been developed and refined by the publicly available programs FIRE, [106], REDUZE, [107], MINCER, [108] and AIR, [109].

Integration by Parts (IBP) Identities

For illustrative purposes, one way of relating Feynman integrals of a given topology to their master integrals is to use *Integration by parts* (IBP) identities [110, 111]. IBP identities stem from the fact that the integral of a total derivative is zero, provided that its integrand is bounded, i.e.

$$\int_{-\infty}^{+\infty} \frac{d^dk}{i\pi^{\frac{d}{2}}} \frac{\partial}{\partial k^\mu} \omega^\mu f(\{k\},\{p\}) = \omega^\mu f|_{-\infty}^{+\infty} = 0\,, \tag{4.47}$$

where ω^μ is an arbitrary four-momentum.

4.5.2 Example

Integration by Parts (IBP) Identities

To provide an idea of how IBP identities can be used in facilitating a reduction to master integrals, consider the simplest possible example, the massive tadpole integral:

$$T^\nu(d) = \int_{-\infty}^{+\infty} \frac{d^dk}{i\pi^{\frac{d}{2}}} \frac{1}{(k^2-m^2)^\nu}\,. \tag{4.48}$$

Applying Eq.(4.47) with $\omega^\mu = k^\mu$:

$$\begin{aligned}
0 &= \int_{-\infty}^{+\infty} \frac{d^dk}{i\pi^{\frac{d}{2}}} \frac{\partial}{\partial k^\mu} \left[\frac{k^\mu}{(k^2-m^2)^\nu}\right] \\
&= \int_{-\infty}^{+\infty} \frac{d^dk}{i\pi^{\frac{d}{2}}} \left[\frac{1}{(k^2-m^2)^\nu}\left(\frac{\partial k^\mu}{\partial k^\mu}\right) - \nu k^\mu \frac{2k_\mu}{(k^2-m^2)^{\nu+1}}\right] \\
&= \delta^\mu_\mu T^\nu(d) - 2\nu \int_{-\infty}^{+\infty} \frac{d^dk}{i\pi^{\frac{d}{2}}} \frac{k^2-m^2+m^2}{(k^2-m^2)^{\nu+1}} \\
&= (d-2\nu)T^\nu(d) - 2\nu T^{\nu+1}(d) \\
\implies T^{\nu+1}(d) &= \frac{d-2\nu}{2\nu m^2} T^\nu(d)\,.
\end{aligned} \tag{4.49}$$

Thus, by application of Eq.(4.47), integrals of the tadpole topology can always be reduced to their simplest form, that of the master integral. Identities such as this provide the method of reducing Feynman integrals to master integrals. However, the reduction can rarely be performed this directly. Realistically, IBP relations contain many terms and need to be solved as a large system of equations. A more realistic example is given by I23, shown in Fig. 4.4. The general Feynman integral corresponding to this diagram is:

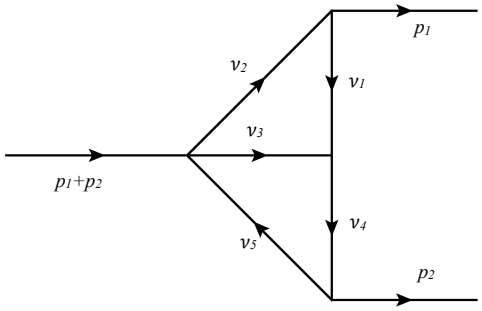


Figure 4.4: A two-loop master integral that contributes to $H \to Z\gamma$ [92], with general powers of propagators.

$$
\begin{aligned}
&I^{\nu_1\nu_2\nu_3\nu_4\nu_5}(d) \\
&= \int \frac{d^dk}{i\pi^{\frac{d}{2}}} \left(\frac{1}{k^2}\right)^{\nu_1} \left(\frac{1}{(k+p_1)^2}\right)^{\nu_2} \left(\frac{1}{(k-l)^2}\right)^{\nu_3} \left(\frac{1}{l^2}\right)^{\nu_4} \left(\frac{1}{(l-p_2)^2}\right)^{\nu_5} \ .
\end{aligned} \tag{4.50}
$$

Applying Eq.(4.47) with $\omega^\mu = k^\mu$ yields, after performing the differentiation:

$$
\begin{aligned}
0 = &\int \frac{d^dk}{i\pi^{\frac{d}{2}}} \left(d - 2\nu_1 - 2\nu_2 \cdot \frac{k\cdot(k+p_1)}{(k+p_1)^2} - 2\nu_3 \cdot \frac{k\cdot(k-l)}{(k-l)^2}\right) \\
&\times I^{\nu_1\nu_2\nu_3\nu_4\nu_5}(d) \, ,
\end{aligned} \tag{4.51}
$$

and rewriting the numerators in terms of propagators;

$$2(k^2 + k\cdot p_1) = k^2 + (k+p_1)^2 - p_1^2 \, , \tag{4.52}$$

$$2(k^2 - k\cdot l) = (k-l)^2 + k^2 - l^2 \ . \tag{4.53}$$

With this, Eq.(4.51) can now be simplified to

$$
\begin{aligned}
0 = &\int \frac{d^Dk}{i\pi^{\frac{D}{2}}} \left((D - \nu_{23} - 2\nu_1) - \nu_2 \cdot \frac{k^2}{(k+p_1)^2} - \nu_3 \cdot \frac{k^2-l^2}{(k-l)^2}\right) \\
&\times I^{\nu_1\nu_2\nu_3\nu_4\nu_5}(d) \ .
\end{aligned} \tag{4.54}
$$

Therefore, the IBP relation reads:

$$
\begin{aligned}
&\frac{\nu_3(I^{\nu_1\nu_2(\nu_3+1)(\nu_4-1)\nu_5}(d) - I^{(\nu_1-1)\nu_2(\nu_3+1)\nu_4\nu_5}(d)) - \nu_2(I^{(\nu_1-1)(\nu_2+1)\nu_3\nu_4\nu_5}(d))}{(d - \nu_2 - \nu_3 - 2\nu_1)} \\
&= I^{\nu_1\nu_2\nu_3\nu_4\nu_5}(d) \ .
\end{aligned} \tag{4.55}
$$

From this it can be seen that in order to perform a reduction of a non-trivial Feynman integral, multiple integration by parts identities are needed. These are then solved as a system of linear equations. This will be further elaborated upon in Chapter 5 concerning the Reduze program and the quasi-finite basis. Before moving on however, it must be noted that the identity of the master integrals is *not* always clear prior to performing the reduction. Sometimes it is more helpful to use tensor rather than scalar master integrals.

4.6 Summary

Finally, it is now clear that the bottleneck to precisely predicting the differential cross section for a particle physics process is the calculation of *master integrals*, from which higher-order corrections can be constructed. As discussed in the introduction, this has led to the development of a wide variety of techniques to do so, both analytic and numerical. It is to subvert this bottleneck in a novel but general way that the TayInt program was developed. The program is an adaption of one of the most established and successful approaches to calculating master integrals, the method of *Sector decomposition* [42–45], starting from a quasi-finite basis [65, 66]. This will be discussed in detail in the following two chapters.

5 | The Quasi-Finite Basis and the Reduze Program

5.1 Introduction

The TayInt program generates approximations for Feynman integrals which are algebraic in the kinematic scales and can be used to produce precise and accurate numerical results quickly at any kinematic point in phase space. This is facilitated by a Taylor expansion at the integrand level. Thus, in order to generate a precise approximation, the Feynman integrands need to be manipulated extensively so that they are in a form optimised for Taylor expanding. Before this process of manipulation can begin, it is best to first start from a form of the integrand that will produce the simplest results when subject to the subsequent manipulation. To be more explicit, the first step (U1) in the conceptual TayInt algorithm (Chapter 9) is to reduce the starting Feynman integral to a quasi-finite basis of Feynman integrals, introduced in Refs. [65, 66], in which all divergences that arise from Feynman parameter integration are removed. The rationale for so doing is that these finite integrals have simpler subsectors, which are generated next, as is explained in detail in the ensuing Chapter 6. In the conceptual TayInt algorithm an automated shell script steers all the necessary jobs in the Reduze [107] program towards the generation of the quasi-finite basis. However, TayInt can produce results just as well for divergent integrals as it can for finite integrals, as will be demonstrated in Chapter 13. Thus, the quasi-finite basis reduction will not be included in the final TayInt algorithm.

5.2 The Reduze Program

Both divergent and quasi-finite Feynman integrals can be derived from a common set of master integrals, the systematic reduction to which is performed by the Laporta algorithm and so by means of IBP and Lorentz invariance [7] identities relations between divergent and quasi-finite integrals may be constructed.

Using the publicly available program Reduze, these reductions to master integrals can be automatically generated. TayInt contains a script which steers the relevant Reduze jobs towards finding a quasi-finite basis for a given divergent Feynman integral. The TayInt program ensures that the divergences are always restricted to the simplest

integral in the quasi-finite basis.

5.2.1 Notation

In order to understand the mechanics of the Reduze program, its notation must first be outlined.

Propagators

As was defined in Chapter 4, inverse propagators $P_{\tilde{j}}$ are linear combinations of the external momenta $p_{\tilde{j}}$, loop momenta k_α and masses $m_{\tilde{j}}$, such that:

$$P_{\tilde{j}}(\{k\},\{p\},m_j^2) = q_{\tilde{j}}^2 - m_{\tilde{j}}^2 + i\delta\,, \tag{5.1}$$

where the $+i\delta$ in Eq. (5.1) results from the solutions of the field equations in terms of causal Green functions and shifts the poles away from the real axis.

Integral Families

An integral family F at the loop level L is a set of n propagators $\{P_1,...,P_n\}$ from which any scalar product (containing at least one loop momentum) from the set of momenta $\{k_1,...,k_L,p_1,...,p_M\}$ can be uniquely expressed. If there are L loop momenta and M external momenta the corresponding integral family must contain $L(L+1)/2 + LM$ propagators, where the first term counts the loop momenta scalar products and the second those between loop and external momenta.

Sectors of Integral Families

Any group of propagators picked from the same integral family form a sector of that integral family. Each different possible sector can be labelled using the formula:

$$ID = \sum_{k=1}^{N} 2^{j_k-1}\,, \tag{5.2}$$

where the sector is composed of propagators $P_{j_1},...,P_{j_N}$ where $N < n$. Using simple combinatorial methods, one sees that in general there are $\binom{n}{N}$ possible sectors of an integral family with N sectors and thus $\sum_{N=0}^{n}\binom{n}{N} = 2^n$ sectors in total, which can be proved inductively.

5.2.2 Indexing Loop Integrals

Numerators of generic scalar loop integrals can be expressed as combinations of inverse propagators, the form of which were given in Eq. 5.1. The powers of each propagator

are denoted as $\nu_{\tilde{j}}$, which can be further subdivided into positive powers $\nu^{+}_{\tilde{j}}$ and negative powers $\nu^{-}_{\tilde{j}}$, allowing a generic scalar loop integral to be cast in the form of linear combinations of integrals of the type:

$$G = \left(\prod_{\alpha=1}^{L} \int \mathrm{d}^{D}\kappa_{\alpha}\right) \frac{1}{[P_{\tilde{j}_1}^{\nu^{-}_{\tilde{j}_1}} ... P_{\tilde{j}_a}^{\nu^{-}_{\tilde{j}_a}}]} \cdot \frac{1}{[P_{\tilde{j}_{a+1}}^{\nu^{+}_{\tilde{j}_{a+1}}} ... P_{\tilde{j}_n}^{\nu^{+}_{\tilde{j}_n}}]} , \tag{5.3}$$

with the propagators explicitly separated according to the sign of their power. This allows the following quantities to be added to F, a and ID as means of labelling a loop integral in Reduze:

$$r = \sum_{\tilde{j}=1}^{a} \nu^{+}_{\tilde{j}} , \tag{5.4}$$

$$s = \sum_{\tilde{j}=1}^{a} \nu^{-}_{\tilde{j}} , \tag{5.5}$$

such that in Reduze a generic scalar loop integral is represented by;

$$G(F, N, ID, r, s, \{\nu_1, ..., \nu_n\}) . \tag{5.6}$$

Herein, F is the integral family, N is the number of propagators in the sector of the family needed to generate the integral, ID is the identification number generated according to Eq. 5.2, r and s give the number of positive and negative propagator powers in the loop integral and $\{\nu_1, ..., \nu_n\}$ are the powers of the propagators in the integral family. The integral with $r = N$ and $s = 0$ is referred to as the *corner* integral of a given sector.

5.3 The Use of Reduze Within TayInt

Now that the Reduze program and its notation has been introduced and explained, its use within the TayInt program will now be explained, with each step of the TayInt shell script that calls the various necessary Reduze jobs being labelled by the abbreviation QFB followed by the step number.

5.3.1 QFB1: Integral Family

The purpose of using Reduze within TayInt is to find a quasi-finite basis for the loop integral the user wishes to calculate, known as the *target integral*. In order to do so, in step QFB1 the corresponding integral family must first be defined, as explained previously (Section 5.2) and the target integral identified and stored in a file `target_integral.txt`. The *target integral* is usually the corner integral of its integral family but not always, owing to the facts that there may be more than one master integral in a given sector and the input to TayInt may be *any* integral that is represented in terms of Feynman parameters.

5.3.2 QFB2: Sector Mappings

To achieve this, in step QFB2 relations need to be derived between the corner integral and integrals corresponding to different sectors using IBP identities, as previously described in Chapter 4. The IBP reduction step is computationally intensive, so to ensure it is only applied to the minimal necessary extent, transformations in the loop momenta known as sector mappings are applied to the corner integral to find possible relations between the corner integral and other integrals within different sectors. There are three pieces of information derived from sector mappings:

1. Zero sectors: the sectors which either contain only integrals which evaluate to zero or do not contain L independent loop momenta are located using sector mappings and removed from consideration.

2. Sector relations: the shifts in momentum which map a given integral onto an integral in a different sector.

3. Sector symmetries: the shifts in momentum which map a given integral onto an integral within the same sector.

This information can then be used to assist the subsequent IBP reductions by eliminating any redundant sectors as early as possible and by finding injective mappings between the corner integral and integrals both within and outside of its sector. The TAYINT shell script generates the sector mappings by calling the relevant REDUZE job, using the command `mpirun -np 5 reduze sectormappings.yaml`. At this stage, the TAYINT shell script will also check if the user has entered an integral which appears within the zero sectors. If so, a warning is returned for the user, giving them the option to quit the program and start again if they have made a mistake. The subject of sector mappings is highly non-trivial and is not the focus of this thesis, for a detailed discussion see [107].

5.3.3 QFB3: Finiteness Search

The next step, QFB3, in the TAYINT shell script is to generate the finite versions of the integrals in the sector, performed by running the reduze job `finite.yaml`. As was shown in Eq. 4.15, a generic scalar L-loop integral is divergent, with degree of divergence given by

$$S_k = L \cdot D - \sum_{j=1}^{N} \nu_{\tilde{j}} \,, \tag{5.7}$$

in D-dimensions. Thus, in order to search for the finite integrals within the given integral family, the `finite.yaml` job alters the the powers of the propagators, $\nu_{\tilde{j}}$ and the number of dimensions, D, within each integral, using the algorithm in [65]. In the former case, integrals free of ultraviolet divergences are found, in the latter, those without divergences of an infrared nature. The search proceeds bottom-up, starting with the minimal number of shifts to the propagator powers and dimensions and proceeds as

far as requested by the user of TayInt. The output is a list of the finite integrals in the family, stored `finite.txt`. At this stage, the TayInt shell script will also check if the user has entered an integral which is already finite and return a warning to the user, giving them the option to quit the program and start again if they have made a mistake. A detailed description of this method is given in Refs. [65, 66].

5.3.4 QFB4: Classifying the Finite Integrals

In step QFB4 the TayInt shell script takes the list of the finite integrals in the family, `finite.txt` and separates it into a list of the finite versions of the integral entered by the user, `finite_targets.txt` and the finite integrals simpler than the target, `finite_auxilliary.txt`.

5.3.5 QFB5: IBP Generation of the Target's Master Integrals

In step QFB5, the IBP identities for the target integral are generated using the job `shift_targets.yaml` and, along with these identities, the master integrals for the target integral are stored in `shift_masters.txt`.

5.3.6 QFB6: IBP Generation of the Target's Finite Integral Basis

All of the integrals in the sector corresponding to the target integral can be expressed in terms of the master integrals, including the finite integrals. The Reduze program provides the option `preferred_masters` in its `reduction.yaml` job and so in step QFB6 the finite integrals found in `finite.txt` and the master integrals and list of necessary IBP identities in `shift_masters` are combined into a single file, `userbasis`, which is used as the `preferred_masters` file. This `reductions.yaml` job will then use the IBP identities to try and express the target integral in terms of the integrals provided as `preferred_masters`. Because both the master integrals and the finite integrals are provided, identities for the target integral and the finite integrals in terms of the master integrals will be generated, allowing Reduze to solve for the target integral in terms for the finite integrals, generating a quasi-finite basis, stored in `finite_output.txt`.

5.3.7 Illustration: QFB1-6

Each of the steps described above are displayed all together to illustrate the process of finding a quasi-finite basis for a loop integral using Reduze jobs within the TayInt algorithm in Fig. 5.1.

5.4 The Application of Reduze Within TayInt

To demonstrate the working of the method described in the previous section and illustrated in Fig. 5.1, the outcome of each step on the two-mass sunrise integral, depicted in Fig. 5.2, will be described next.

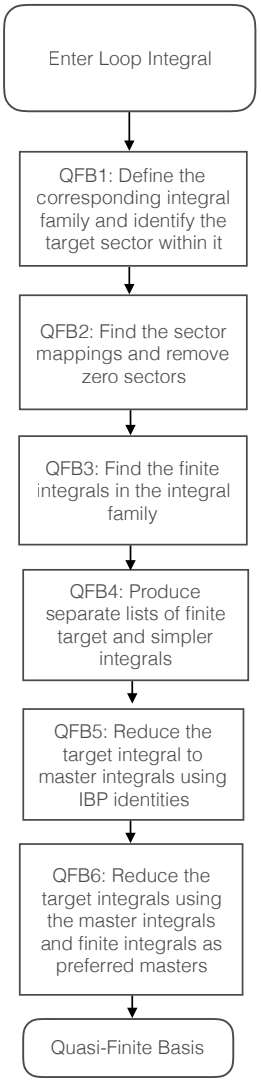


Figure 5.1: The relationship between the steps in the TayInt shell script used to steer Reduze jobs towards the generation of a quasi-finite basis part for a generic scalar loop integral.

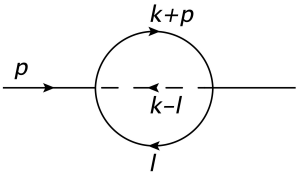


Figure 5.2: The two-loop divergent sunrise $S14^{01110}$.

5.4.1 QFB1: Integral Family

The two-mass sunrise integral (target integral), as shown in Fig. 5.2, is contained within the integral family, defined in the file `integralfamilies.yaml`:

Example 5.4.1

```
name:  "S14"
loop_momenta:  [k,l]
propagators:
- [ "k", m ]
- [ "k-l", 0 ]
- [ "k+p", m ]
- [ "l", m ]
- [ "l+p", m ] .
```

The sunrise integral corresponds to the sector containing propagators two,three and four each with unit powers, so in Reduze notation it is denoted as:

Example 5.4.2

$$G(S14, 3, 14, 3, 0, \{0, 1, 1, 1, 0\}) ,$$

wherein $N = 3$, the number of propagators in the sector, $ID = 2^1 + 2^2 + 2^3 = 14$, $r = 3$ and $s = 0$. As $r = 3$ and $s = 0$, it is clear that this is the corner integral of its sector. Its name will be shortened to $S14^{01110}$ for brevity where necessary.

5.4.2 QFB2: Sector Mappings

For the two-mass sunrise integral, depicted in Fig. 5.2, some examples of each type of information returned by the sector mappings job are:

1. Zero sectors: Sectors such as $S14^{1000)}$ and $S14^{10100}$ vanish because they only contain one loop momentum, k and the sector $S14^{01000}$ is zero because it is scaleless. Hence these and the other such sectors are removed from consideration immediately. so as not to apply IBP reduction to more integrals than necessary.

2. Sector relations: an example of a sector relation is given by `[[S14, 5], [[k, 2k-l], [l, k-l]]]`, which means that there exists a mapping from the sunrise to an integral within another sector generated by $l \to k$ and $k - l \to 2k - l$.

3. Sector symmetries: an example of a sector symmetry for sector $S14$ is given by `- [[k, k-l], [l, -l]]`, which means that the sectors of $S14$ are unchanged by

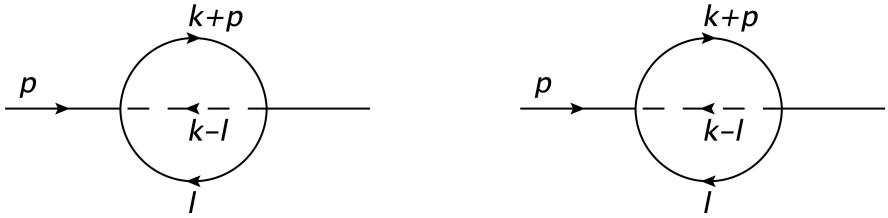


Figure 5.3: The sector symmetry of the divergent sunrise $S14^{01110}$.

the exchanges $k \leftrightarrow k-l, l \leftrightarrow -l$, as demonstrated using the sunrise integral in Fig. 5.3.

5.4.3 QFB3: Finiteness Search

In the case of the sunrise diagram, the `finite.yaml` job returns the following integrals in the list `finite.txt`:

Example 5.4.3

$$
\begin{aligned}
&G(S14,3,14,5,0,\{0,1,2,2,0\})\,,\\
&G(S14,3,14,6,0,\{0,1,3,2,0\})\,,\\
&G(S14,3,14,6,0,\{0,1,2,3,0\})\,,\\
&G(S14,2,5,6,0,\{3,0,3,0,0\})\,,\\
&G(S14,2,5,7,0,\{4,0,3,0,0\})\,,\\
&G(S14,2,5,7,0,\{3,0,4,0,0\})\,,\\
&G(S14,2,9,6,0,\{3,0,0,3,0\})\,,\\
&G(S14,2,9,7,0,\{4,0,0,3,0\})\,,\\
&G(S14,2,9,7,0,\{3,0,0,4,0\})\,,\\
&G(S14,2,12,6,0,\{0,0,3,3,0\})\,,\\
&G(S14,2,12,7,0,\{0,0,4,3,0\})\,,\\
&G(S14,2,12,7,0,\{0,0,3,4,0\})\,,
\end{aligned}
$$

if the user sets the limit on the sum of the modulus of the propagator powers to $r = 7$. As the sunrise contains only ultraviolet divergences, only shifts in the propagator powers are employed as part of the search for finite integrals within the family.

5.4.4 QFB4: Classifying the Finite Integrals

The first three integrals are finite versions of the sunrise integral and so are stored in a separate list, `finite_targets.txt`, while the remaining integrals are all simpler, so are stored in `finite_auxilliary.txt`.

5.4.5 QFB5: IBP Generation of the Target's Master Integrals

In the case of the sunrise integral, the target integral *is* a master integral, so only the remaining master integrals need to be generated in step QFB5 and stored in `shift_masters.txt`, which reads:

Example 5.4.4

$$
\begin{aligned}
&G(S14, 3, 14, 4, 0, \{0, 1, 1, 1, 0\}) \,, \\
&G(S14, 3, 14, 4, 0, \{0, 1, 1, 2, 0\}) \,, \\
&G(S14, 3, 14, 4, 0, \{0, 2, 1, 1, 0\}) \,, \\
&G(S14, 3, 14, 4, 0, \{0, 1, 2, 1, 0\}) \,.
\end{aligned}
$$

5.4.6 QFB6: IBP Generation of the Target's Finite Integral Basis

In step QFB6, the finite integrals and master integrals are combined into the same file, `userbasis`, which contains the integrals specified in `finite.txt`, described in Subsection 5.4.3. Using `userbasis` as the source of preferred masters in the Reduze job `reductions.yaml` generates results for all the master integrals in terms of the finite integrals, which read:

$$
\begin{aligned}
\mathrm{S14}^{01110} = &\frac{8\,m^2\,(p^2-4\,m^2)\,(p^2+2\,m^2)}{(-3+D)(-8+3\,D)(-10+3\,D)} \cdot \mathrm{S14}^{01320} \\
&+ \frac{((4-D)\,p^4+(-5+D)\,8\,m^4+(18-5\,D)\,4\,p^2m^2)}{(-3+D)(-8+3\,D)(-10+3\,D)} \cdot \mathrm{S14}^{01220} \\
&- \frac{16\,m^4\,((-4+D)\,p^2+2\,(-24+7\,D)\,m^2)}{(-3+D)(-4+D)^2(-8+3\,D)(-10+3\,D)} \cdot \mathrm{S6}^{30300} \,,
\end{aligned}
\tag{5.8}
$$

$$
\begin{aligned}
\mathrm{S14}^{02110} = &\frac{8\,m^2\,(-p^2+4\,m^2)}{(-4+D)(-10+3\,D)} \cdot \mathrm{S14}^{01320} \\
&+ \frac{(p^2-10m^2)}{(-10+3\,D)} \cdot \mathrm{S14}^{01220} \\
&- \frac{16\,m^4}{(-4+D)^2(-10+3\,D)} \cdot \mathrm{S6}^{30300} \,,
\end{aligned}
\tag{5.9}
$$

$$
\begin{aligned}
\mathrm{S14}^{01210} = &\frac{4\,m^2\,(p^2-4\,m^2)}{(-3+D)(-10+3\,D)} \cdot \mathrm{S14}^{01320} \\
&+ \frac{2Dm^2-3p^2+Dp^2-10m^2}{(-3+D)(-10+3\,D)} \cdot \mathrm{S14}^{01220} \\
&- \frac{16(-7+2D)\,m^4}{(-4+D)^2(-3+\,D)(-10+3\,D)} \cdot \mathrm{S6}^{30300} \,,
\end{aligned}
\tag{5.10}
$$

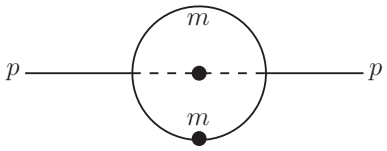


Figure 5.4: The two-loop finite sunrise $\mathrm{S14}^{01220}$.

$$\begin{aligned}\mathrm{S14}^{01120} = &\frac{4\,m^2\,(p^2-4\,m^2)}{(-3+D)(-10+3\,D)}\cdot\mathrm{S14}^{01320}\\ &+\frac{2Dm^2-3p^2+Dp^2-10m^2}{(-3+D)(-10+3\,D)}\cdot\mathrm{S14}^{01220}\\ &-\frac{16(-7+2D)\,m^4}{(-4+D)^2(-3+\,D)(-10+3\,D)}\cdot\mathrm{S6}^{30300}\,,\end{aligned}\tag{5.11}$$

from which the first relation is selected and stored in `finite_output.txt`. At the end of step QFB6, the quasi-finite basis for the divergent sunrise has now been found. Each increased propagator power is represented by a dot in the Feynman diagram, as shown in Fig. 5.4 for $\mathrm{S14}^{01220}$.

5.5 Summary

At the end of this chapter, the production of a quasi-finite basis for Feynman integrals, through use of IBP identities together with the concept of shifted dimensions and powers of propagators within the Reduze program has been demonstrated. The method of collecting Reduze jobs into a script employed by the TayInt program to produce such a basis has also been outlined and and illustrated with an example. Thus, step U1 in the conceptual TayInt algorithm has been completed, as shown by the corresponding bold label in Table 5.1 below. Now that the method for reducing a loop integral to a quasi-finite basis has been demonstrated, the next step is to decompose the integrals in that basis into subsectors which disentangle their parameters further, using the method of sector decomposition, which will be introduced in the following chapter.

Table 5.1: Summary of the individual steps of the conceptual TayInt algorithm, with the labels of the step U1 presented so far typeset in boldface.

U1: reduce the Feynman Integral to a quasi-finite basis	
U2: perform a sector decomposition on the finite integrals in the basis	
Below threshold	Over threshold
BT1: $t_j \to y_j$	OT1: $t_j \to \theta_j$, generate $\mathcal{K}$
BT2: Taylor expand the integrand and integrate	OT2: find optimum $\Theta_{o(0),\ldots,o(J-1)}$
	OT3: perform one-fold integrations
	OT4: post-integration, find optimum $\Theta_{o(0),\ldots,o(J-2)}$
	OT5: determine partition $\mathcal{P}_j$
	OT6: Taylor expand and integrate

6 | Sector Decomposition

6.1 Introduction

Sector decomposition [42–45] splits up a Feynman integral such that the structure of each resultant subsector integrand has its Feynman parameters structure maximally separated. This allows the most efficient conversion of a dimensionally regulated loop integral into a Laurent series, a concept that was discussed in Chapter 4. This approach has been embedded in publicly available programs [46–55] culminating in successful phenomenological applications up to two-loop five-point four-scale processes [34, 56–60, 60–64].

However, in the TayInt program, sector decomposition is used because each iterated subsector has its *threshold singularities* (which do not depend on the variables of integration but rather on the kinematic scales) more spaced out than the full Feynman integral. Thus, it is easier to avoid the threshold singularities with the final TayInt algorithm.

The method of sector decomposition is therefore very important to ease the workload of the over-threshold part of the final TayInt algorithm and so is formally introduced step by step in Section 6.2 and fully illustrated in Section 6.3.

6.2 The Method of Sector Decomposition

To best prepare master Feynman integrals for a Taylor expansion in regions of phase space reached by crossing thresholds, it is easier to work with their subsectors. An algorithmic procedure can then be applied to find a contour configuration for each subsector that produces a smooth integrand. This is achieved by use of sector decomposition, using the code from SecDec version 3.0.9.

Starting from the generic Feynman integral G given in Eq. (4.41), the method of sector decomposition now proceeds as follows.

6.2.1 SD1: Splitting into Hierarchies

The Feynman integrands which the TayInt program aims to calculate have fully overlapping polynomial structures in all of their Feynman parameters. To separate these, the first step is to break the integral up into N primary sector integrals, where N is the

number of Feynman parameters, by inserting:

$$1 = \sum_{l=1}^{N} \prod_{\substack{\tilde{j}=1 \\ l \neq \tilde{j}}} \Theta(t_l - t_{\tilde{j}}) \,, \tag{6.1}$$

where $\Theta(x)$ is the Heaviside function [88].

In each primary sector G_l^{P}, t_l is the largest Feynman parameter, termed the *primary* Feynman parameter in each case. Thus the effect of the primary sector decomposition is to split the Feynman integral into N primary sectors. Within each primary sector the corresponding Feynman parameter is the largest, such that:

$$G = \frac{(-1)^{N_\nu}}{\prod_{\tilde{j}=1}^{N} \Gamma(\nu_{\tilde{j}})} \sum_{l=1}^{N} G_l^{\mathrm{P}} \ . \tag{6.2}$$

6.2.2 SD2: Remapping the Feynman Parameters

For each such hierarchy, in the next step the largest Feynman parameter is then isolated within the Dirac delta function so that it can be integrated out. To perform the isolation of each primary Feynman parameter, the following transformation logic: $t \to \tau$, between the Feynman parameter space of the full Feynman integrand and that of each primary sector, is implemented;

$$t_{\tilde{j}}^{(l)} = \begin{cases} t_l \tau_{\tilde{j}} \,, & \text{for } \tilde{j} < l \\ t_l \,, & \text{for } \tilde{j} = l \\ t_l \tau_{\tilde{j}} \,, & \text{for } \tilde{j} > l \end{cases} \tag{6.3}$$

such that $\vec{t}^{(l)}$ is the vector of Feynman parameters for each primary subsector l and $\tilde{j}$ labels its entries, which run from 1 to the number of Feynman parameters, N. This transformation can also be written as:

$$t_{\tilde{j}}^{(l)} = t_l \tau_{\tilde{j}} \Theta(l - \tilde{j}) + t_l \delta_{\tilde{j}l} + t_l \tau_{\tilde{j}} \Theta(\tilde{j} - l) \ . \tag{6.4}$$

At the end of this step, the Feynman parameters have been remapped within each primary sector to make the hierarchy imposed in SD1 explicit.

6.2.3 SD3: Integration of the Primary Feynman Parameter

Using this notation, in step SD3 the integration within each primary sector of the Feynman integral can be simplified so that it is exclusively over the remapped Feynman parameters, $\tau_{\tilde{j}}$, using that:

$$\int_0^\infty d\tau_{\tilde{j}} \Theta(t_l(1 - \tau_{\tilde{j}}) = \int_0^1 d\tau_{\tilde{j}} \ . \tag{6.5}$$

Using the substitution:

$$u_l = t_l \left(1 + \sum_{\tilde{j}=1}^{N-1} \tau_{\tilde{j}} \right) , \tag{6.6}$$

the primary Feynman parameter in Eq. 6.5 may be integrated out as:

$$\begin{aligned} &\int_0^\infty \frac{dt_l}{t_l} \delta \left(1 - t_l \left(1 + \sum_{\tilde{j}=1}^{N-1} \tau_{\tilde{j}} \right) \right) \\ &= \int_0^\infty \frac{du_l \left(1 + \sum_{\tilde{j}=1}^{N-1} \tau_{\tilde{j}} \right)}{u_l \left(1 + \sum_{\tilde{j}=1}^{N-1} \tau_{\tilde{j}} \right)} \delta \left(1 - u_l \right) = \int_0^\infty \frac{du_l}{u_l} \delta \left(1 - u_l \right) = 1 \ . \end{aligned} \tag{6.7}$$

In the above equation, t_l refers to the primary Feynman parameter of each primary sector. At the end of this step, the primary Feynman parameter has been integrated out within each primary sector.

6.2.4 SD4: Writing Down the Primary Sectors

Step SD4 is then to rewrite the variables of integration in terms of the original Feynman variables $\tau_{\tilde{j}} \to t_{\tilde{j}}$, leading to the simplified and distinct *primary sectors* of the Feynman integral at the end of this step.

6.2.5 SD5: Iterated Sector Decomposition

As the Feynman integrals the TayInt program aims to calculate contain maximally overlapping Feynman parameter structures, they are not fully disentangled after a primary sector decomposition. Thus, in step SD5 the steps SD1-2 are applied once again (SD3 is not needed as there is no longer a delta function to be integrated). This produces the iterated subsectors, G^{I}_{lk}.

6.2.6 SD6: Extract Distinct Subsectors

The final step, SD6, is to sum all these iterated subsectors and each distinct term that remains after summation is then relabelled to compose the G^S_l, the final subsectors, which take the general form:

$$G^{\mathrm{S}}_l(\{q\},\{m\}) = \prod_{\tilde{j}=1}^{N-1} \int_0^1 \mathrm{d}t_{\tilde{j}} \, t_{\tilde{j}}^{A_{l\tilde{j}} - B_{l\tilde{j}}\epsilon} \frac{\mathcal{U}_l^{N_\nu - (L+1)D/2}\left(\vec{t}_{\tilde{j}}\right)}{\mathcal{F}_l^{N_\nu - LD/2}\left(\vec{t}_{\tilde{j}}, \{q\}, \{m\}\right)}, \quad l = 1, \ldots, r \ , \tag{6.8}$$

where r is the number of subsector integrals. $A_{l\tilde{j}}$ and $B_{l\tilde{j}}$ are numbers independent of the dimensional regulator ϵ. There are only $N-1$ integrations in total because one

Feynman parameter t_l is always integrated out with the δ-distribution. By construction, the deterministic algorithm results in integrands of the type:

$$\mathcal{U}_l = 1 + u\left(\vec{t}_{\tilde{j}}\right) , \tag{6.9}$$

$$\mathcal{F}_l = s_1 + \sum_\beta s_\beta f_\beta\left(\vec{t}_{\tilde{j}}\right) , \tag{6.10}$$

where $u\left(\vec{t}_{\tilde{j}}\right)$ and $f_\beta\left(\vec{t}_{\tilde{j}}\right)$ are polynomials in the Feynman parameters $t_{\tilde{j}}$ and $s_1, s_\beta \in \{\{q\},\{m\}\}$ are kinematic invariants including masses. If the integral is not finite (which it usually is), the singular behaviour is now contained entirely in the exponents $A_{l\tilde{j}}$ of Eq. (6.8). As the integrals to be computed with TayInt are typically finite, a sector decomposition might seem unnecessary. However, it is observed to be vital for an improved convergence of the Taylor expansion.

As the primary Feynman parameter has been integrated out, to have a sensible hierarchy of parameters $t_{\tilde{j}} \to t_j$, where j runs from 0 to $N-1$. At the end of this step the full Feynman integral can then be written in terms of its final subsectors G_l^{S}:

$$G(\{q\},\{m\}) = \frac{(-1)^{N_\nu}}{\prod_{j=1}^N \Gamma(\nu_j)} \Gamma(N_\nu - LD/2) \sum_{l=1}^r G_l^{\mathrm{S}}(\{q\},\{m\}) , \tag{6.11}$$

which have a fully disentangled Feynman parameter structure.

6.3 Sector Decomposition of a Finite Sunrise Integral

To demonstrate the above method (SD1-6), consider the integral $\mathrm{S14}^{01220}$ produced in the quasi-finite basis representation of the divergent sunrise $\mathrm{S14}^{01110}$ (see Fig. 4.3). The Symanzik polynomials were found in Eqs. (4.43) and (4.45) and can be inserted into 4.41 to write down the Feynman integral for $\mathrm{S14}^{01220}$, as

$$\begin{aligned} G =& (-1) \int_0^\infty \int_0^\infty \int_0^\infty dt_1 dt_2 dt_3 \; t_2 t_3 \, \delta(1 - t_1 - t_2 - t_3) \\ & \times (t_1 t_3 + t_2 t_3 + t_1 t_2)^{-1} \left((t_1 t_3 + t_2 t_3 + t_1 t_2)(t_2 + t_3) m^2 - t_1 t_2 t_3 p^2\right)^{-1} , \end{aligned} \tag{6.12}$$

using $N = 3$, $\nu_1 = 1$, $\nu_2 = 2$, $\nu_3 = 2$, $L = 2$ and $D = 4$. This integrand has all three Feynman parameters entangled, depicted (after integrating out t_3) in Fig. 6.1.

6.3.1 SD1: Splitting into Hierarchies

In order to separate them, the first step is to break the $\mathrm{S14}^{01220}$ integral up into N primary sector integrals, where N is the number of Feynman parameters, using Eq. (6.1), so that:

$$\begin{aligned} 1 =& \Theta(t_1 - t_2)\,\Theta(t_1 - t_3) , \\ &+ \Theta(t_2 - t_1)\,\Theta(t_2 - t_3) , \\ &+ \Theta(t_3 - t_1)\,\Theta(t_3 - t_2) . \end{aligned} \tag{6.13}$$

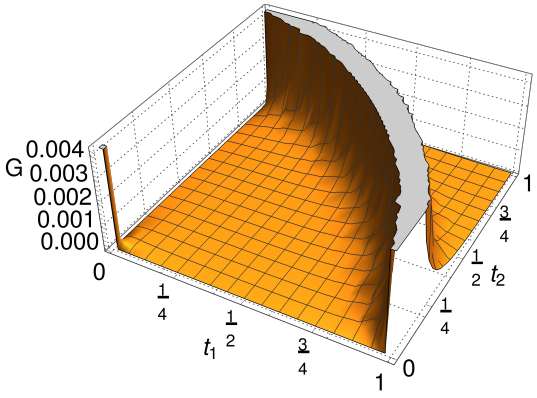


Figure 6.1: The $\mathrm{S14}^{01220}$ integrand at $\mathcal{O}(\epsilon^0)$, setting $p^2 = -\frac{1}{2}m^2$, $m^2 = 20000$ GeV^2.

At the end of this step, the $\mathrm{S14}^{01220}$ integral has been split up into three primary sectors within which t_1, t_2, t_3 are the primary Feynman parameters respectively.

6.3.2 SD2: Remapping the Feynman Parameters

In the next step, applying Eq. (6.3) in the case of $\mathrm{S14}^{01220}$, the following substitutions are generated:

$$\begin{aligned} G_1^{\mathrm{P}} &: t_1 = t_1,\ t_2 = t_1\tau_1,\ t_3 = t_1\tau_2\,, \\ G_2^{\mathrm{P}} &: t_1 = t_2\tau_1,\ t_2 = t_2,\ t_3 = t_2\tau_2\,, \\ G_3^{\mathrm{P}} &: t_1 = t_3\tau_1,\ t_2 = t_3\tau_2,\ t_3 = t_3\ . \end{aligned} \tag{6.14}$$

These give rise to the Feynman parameter space of each primary sector, making the primary Feynman paramater explicit. Thus $\mathrm{S14}^{01220}$ can be decomposed as:

$$\begin{aligned} G = {} & G^{\mathrm{P}}(t_1 \to t_1,\ t_2 \to t_1\tau_1,\ t_3 \to t_1\tau_2)\cdot\Theta(t_1 - t_2)\,\Theta(t_1 - t_3) \\ & + G^{\mathrm{P}}(t_1 \to t_2\tau_1,\ t_2 \to t_2,\ t_3 \to t_2\tau_2)\cdot\Theta(t_2 - t_1)\,\Theta(t_2 - t_3) \\ & + G^{\mathrm{P}}(t_1 \to t_3\tau_1,\ t_2 \to t_3\tau_2,\ t_3 \to t_3)\cdot\Theta(t_3 - t_1)\,\Theta(t_3 - t_2)\ , \end{aligned} \tag{6.15}$$

ready for the integration of the primary Feynman parameter by the end of this step.

6.3.3 SD3: Integration of the Primary Feynman Parameter

The integration of the primary Feynman parameter in the first primary sector of $S14^{01220}$ is performed as follows:

$$\begin{aligned}
G_1^{\mathrm{P}} &= -\int_0^\infty dt_1 \int_0^\infty d\tau_1 \Theta(t_1(1-\tau_1))t_1 \int_0^\infty d\tau_2 \Theta(t_1(1-\tau_2))t_1 \\
&\quad \times \delta(1-t_1(1+\tau_1+\tau_2))(t_1^2\tau_1\tau_2) \times (t_1)^{-2}(\tau_1\tau_2+\tau_1+\tau_2)^{-1} \\
&\quad \times (t_1)^{-3}[-p^2\tau_1\tau_2 + m^2(\tau_1+\tau_2)(\tau_1\tau_2+\tau_1+\tau_2)]^{-1} \\
&= -\int_0^1 d\tau_1 \int_0^1 d\tau_2 \int_0^\infty dt_1 \frac{\delta(1-t_1(1+\tau_1+\tau_2))}{t_1} \\
&\quad \times (\tau_1\tau_2) \times (\tau_1\tau_2+\tau_1+\tau_2)^{-1} \\
&\quad \times [-p^2\tau_1\tau_2 + m^2(\tau_1+\tau_2)(\tau_1\tau_2+\tau_1+\tau_2)]^{-1} \\
&= -\int_0^1 d\tau_1 \int_0^1 d\tau_2 (\tau_1\tau_2) \times (\tau_1\tau_2+\tau_1+\tau_2)^{-1} \\
&\quad \times [-p^2\tau_1\tau_2 + m^2(\tau_1+\tau_2)(\tau_1\tau_2+\tau_1+\tau_2)]^{-1}\,,
\end{aligned} \tag{6.16}$$

making use of the identities given in Eqs. (6.6) and (6.7). Analogously to 6.16, at the end of this step the second and third primary sectors reduce to the simplified form:

$$\begin{aligned}
G_2^{\mathrm{P}} = G_3^{\mathrm{P}} &= -\int_0^1 d\tau_1 \int_0^1 d\tau_2 \; \tau_2(\tau_1\tau_2+\tau_1+\tau_2)^{-1} \\
&\quad \times [-p^2\tau_1\tau_2 + m^2(1+\tau_2)(\tau_1\tau_2+\tau_1+\tau_2)]^{-1}\,.
\end{aligned} \tag{6.17}$$

6.3.4 SD4: Writing Down the Primary Sectors

In the next step it is now a simple matter to rewrite the variables of integration in terms of the original Feynman parameters and complete the decomposition of $S14^{01220}$ into its primary sectors.

Comparing the form of Eqs. (6.16) and (6.17) to that of Eq. (6.12), it is clear that at the end of this step several of the overlapping Feynman parameters have been separated, as the transition:

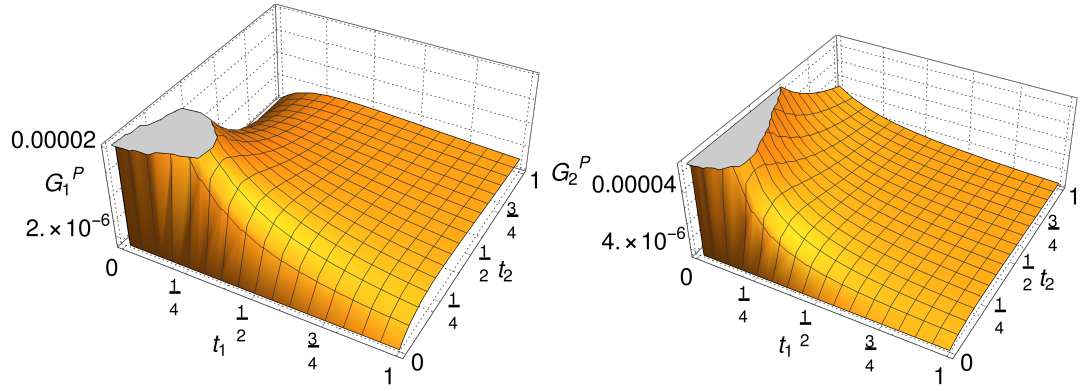
$$(t_1t_3+t_2t_3+t_1t_2)^{-1} \to (t_1t_2+t_1+t_2)^{-1}\,, \tag{6.18}$$

demonstrates. However, they are not fully disentangled, as can be seen in the limit $t_1, t_2 \to 0$ from consulting the plots in Fig. 6.2(a) and 6.2(b).

6.3.5 SD5: Iterated Sector Decomposition

Thus, the next step is to repeat steps SD1-4 for the primary sectors given in Eqs. (6.16) and (6.17), generating the iterated subsectors, G_{lk}^{I}. As each primary sector of $S14^{01220}$ only has two remaining integration parameters, t_1 and t_2, the only possible decomposition of their Feynman parameter space is given by:

$$1 = \Theta\left(t_1 - t_2\right) + \Theta\left(t_2 - t_1\right)\,, \tag{6.19}$$



(a) (b)

Figure 6.2: The two distinct primary sectors of the S14^{01220} integrand at $\mathcal{O}(\epsilon^0)$, (a) G_1^{P} and (b) G_2^{P}, setting $p^2 = -\frac{1}{2}m^2$, $m^2 = 20000$ GeV^2.

which is inserted into each, generating two iterated subsectors for each primary sector. In the first iterated subsector, the substitution $t_1 = t_1, t_2 = t_1\tau_1$ is carried out and in the second, $t_1 = t_2\tau_1, t_2 = t_2$. As G_1^{P} is symmetric in t_1 and t_2, both iterated subsectors are identical and are equal to:

$$\begin{aligned} G_{11}^{\mathrm{I}} = G_{12}^{\mathrm{I}} &= -\int_0^1 dt_1 \int_0^1 d\tau_1 t_1^3 \cdot t_1^{-3}\tau_1 \times (1+\tau_1+t_1\tau_1)^{-1} \\ &\quad \times [-p^2\tau_1 + m^2(1+\tau_1)(1+\tau_1+t_1\tau_1)]^{-1} \\ &= -\int_0^1 dt_1 \int_0^1 dt_2 t_2 \times (1+t_2+t_1t_2)^{-1} \\ &\quad \times [-p^2 t_2 + m^2(1+t_2)(1+t_2+t_1t_2)]^{-1}\,, \end{aligned} \tag{6.20}$$

performing the relabelling $\tau_1 \to t_2$ in the second step. However G_2^{P} is not symmetric in t_1 and t_2, so the two iterated sectors it gives rise to will be distinct and identical to those that emanate from G_3^{P}. They read:

$$\begin{aligned} G_{21}^{\mathrm{I}} &= -\int_0^1 dt_1 \int_0^1 dt_2\ t_2(1+t_2+t_1t_2)^{-1} \\ &\quad \times [-p^2 t_1 t_2 + m^2(1+t_1t_2)(1+t_2+t_1t_2)]^{-1}\,, \end{aligned} \tag{6.21}$$

$$\begin{aligned} G_{22}^{\mathrm{I}} &= -\int_0^1 dt_1 \int_0^1 dt_2\ (1+t_1+t_1t_2)^{-1} \\ &\quad \times [-p^2 t_1 t_2 + m^2(1+t_2)(1+t_1+t_1t_2)]^{-1}\,, \end{aligned} \tag{6.22}$$

At the end of this step the integral S14^{01220} has been split into pieces which have their a fully disentangled Feynman parameter structure.

6.3.6 SD6: Extract Distinct Subsectors

Finally, in step SD6, the distinct subsectors are extracted. In the case of $S14^{01220}$, these final subsectors G_l^S are

$$\begin{aligned} G_1^S &= G_{11}^I + G_{12}^I \\ &= -2\int_0^1 dt_1 \int_0^1 dt_2 t_2 \times (1+t_2+t_1t_2)^{-1} \\ &\times [-p^2 t_2 + m^2(1+t_2)(1+t_2+t_1t_2)]^{-1}\,, \end{aligned} \tag{6.23}$$

$$\begin{aligned} G_2^S &= G_{21}^I + G_{31}^I \\ &= -2\int_0^1 dt_1 \int_0^1 dt_2\ t_2(1+t_2+t_1t_2)^{-1} \\ &\times [-p^2 t_1t_2 + m^2(1+t_1t_2)(1+t_2+t_1t_2)]^{-1}\,, \end{aligned} \tag{6.24}$$

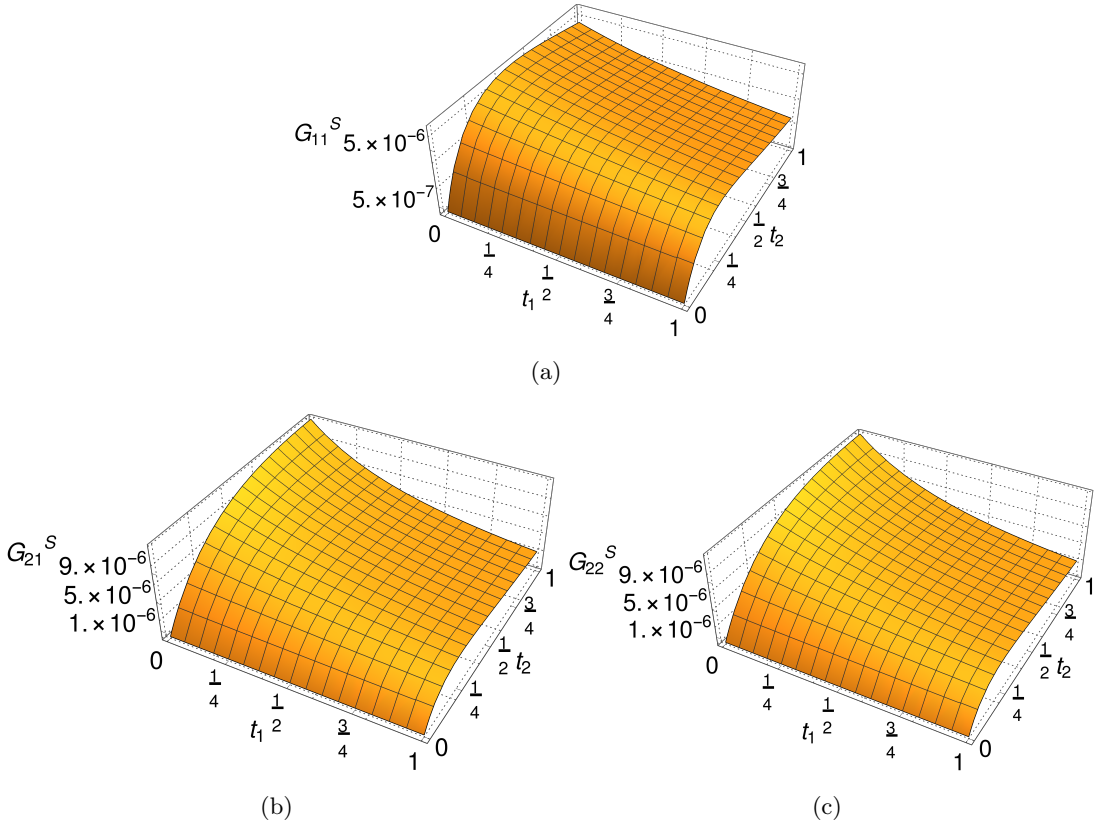
$$\begin{aligned} G_3^S &= G_{22}^I + G_{32}^I \\ &= -2\int_0^1 dt_1 \int_0^1 dt_2\ (1+t_1+t_1t_2)^{-1} \\ &\times [-p^2 t_1t_2 + m^2(1+t_2)(1+t_1+t_1t_2)]^{-1}\ . \end{aligned} \tag{6.25}$$

At the end of this step the Feynman parameters of $S14^{01220}$ are now fully disentangled, as can be seen from Fig. 6.3.

6.4 Summary

In this chapter, the way to simplify the Feynman paramater structure of master integrals has been introduced, explained and illustrated. It is done by means of the general method of Sector decomposition. This method is needed for the TayInt program because Feynman integrals contain threshold singularities which do not depend on the integration variables. Sector decomposition cannot always exclude these from the region of integration but it does organise them as much as possible. This makes them easier to avoid by a change of contour configuration, as will be discussed in Chapter 9. The rationale behind this is that the disentanglement of the singularities in Feynman parameter space spreads out the threshold singularities within the integrand. This effect was demonstrated for the sunrise integral in this chapter.

These threshold singularities independent of the variables of integration will be discussed in the next chapter. At the end of this chapter, steps U2 in the conceptual and U1 in the final TayInt algorithms have been filled in, as shown by the boldface for the corresponding labels in tables 6.1, 6.2 below.



(a)

(b) (c)

Figure 6.3: The S14^{01220} integrand at $\mathcal{O}(\epsilon^0)$ is shown decomposed into its three distinct final subsectors, (a) G_1^{S}, (b) G_2^{S}, (c) G_3^{S}, setting $p^2 = -\frac{1}{2}m^2$, $m^2 = 20000\ \mathrm{GeV}^2$.

Table 6.1: Summary of the individual steps of the conceptual TayInt algorithm, with the labels of the steps U1 and U2 presented so far typeset in boldface.

U1: reduce the Feynman Integral to a quasi-finite basis	
U2: perform a sector decomposition on the finite integrals in the basis	
Below threshold	Over threshold
BT1: $t_j \to y_j$	OT1: $t_j \to \theta_j$, generate $\mathcal{K}$
BT2: Taylor expand the integrand and integrate	OT2: find optimum $\Theta_{o(0),\ldots,o(J-1)}$
	OT3: perform one-fold integrations
	OT4: post-integration, find optimum $\Theta_{o(0),\ldots,o(J-2)}$
	OT5: determine partition $\mathcal{P}_j$
	OT6: Taylor expand and integrate

Table 6.2: Summary of the individual steps of the final TayInt algorithm, with the labels of the step U1 presented so far typeset in boldface.

U1: perform a sector decomposition on the finite integrals in the basis	
U2: locate all the thresholds of the Feynman integral	
Below threshold	Over threshold
BT1: $t_j \to y_j$	OT1: generate partition sets, full and partial contour configurations, kinematic training and cross-validation sets
BT2: Taylor expand the integrand and integrate	OT2: find partition:plain ratios for the full and partial contours
	OT3: Choose optimal full and partial contour by negative exclusion, backup with positive selection
	OT4: Perform kinematic cross validation
	OT5: Assess the optimal full and partial contours and choose between them
	OT6: Generate the uniform partition ratios on the chosen contours
	OT7: Assess the accuracy of the chosen contours to decide if a high-uniform partitioning can be used for all subsectors
	OT8: If not, use the accuracy and improvement assessment to decide if a low- or high-uniform partitioning is appropriate
	OT9: If not, analyse the subsectors on a per-variable basis and find the optimal varied partitioning
	OT10: Taylor expand and integrate

7 | The Landau Conditions

7.1 Introduction

In order to compute a differential cross section it is essential to understand the analyticity properties of its constituent Feynman integrals and how these are influenced by the singularities of their integrands in the space of external variables which are not integrated over, known as *threshold singularities*. These singularities arise when the propagators of the Feynman integral are on-shell, producing a zero in the integral's denominator. However the locations of these singularities cannot be expressed as the poles of the propagators, because the propagators also contain loop momenta, which are integrated over. Thus the only meaningful way to express the locations of these singularities is in terms of the external kinematic scales in the Feynman integral, p_i.

7.2 Analyticity of Integrals

As the TayInt program aims to provide algebraic approximations for arbitrary Feynman integrals, it must be able to manipulate the integrand of a Feynman integral so that these approximations can still be produced even when multiple threshold singularities are present. But before considering Feynman integrals, it is instructive to examine the properties of a one-variable integral of an analytic function $f(z)$:

$$f(z) = \int_{c_{ab}} g(z, \alpha) d\alpha \,, \tag{7.1}$$

in which the integrand $g(z, \alpha)$ contains z-dependent singularities. The analyticity of $f(z)$ on a domain $\Xi \subset \mathbb{C}$ is determined by whether or not the integration path $c_{ab} \subset \Xi$ can be smoothly deformed away from these peripatetic singularities. For example, referring to Fig. 7.1, the integration path, c_{ab}, does not intersect any singularities for $z = z_0$. But, moving from $z = z_0$ (left) to to $z = z_1$ (middle) leads to the singularity $\alpha_1(z_1)$ cutting the integration path c_{ab}. However this does not mean that $f(z)$ is non-analytic. The rightmost figure of 7.1 shows that the integration path c_{ab} can be smoothly deformed so that when $z = z_0 \to z_1$ the singularities of $g(z, \alpha)$ do not touch it. This smooth deformation to avoid singularities dependent on external variables is known as an analytic continuation, introduced in Chapter 3 and used to regularise ultraviolet and infrared

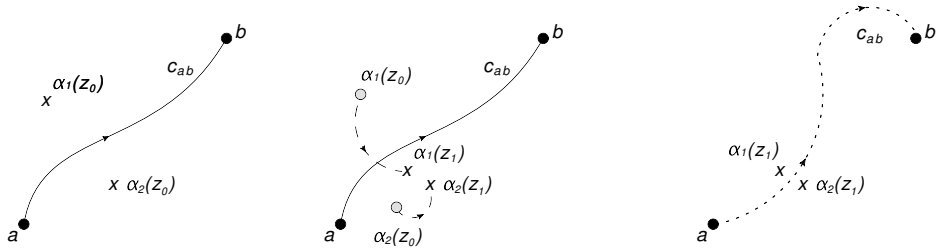


Figure 7.1: From left to right: a path c_{ab} for computing the integral in Eq. (7.1), the integrand of which has singularities at α_1 and α_2 which depend on the external parameter z, when the value of z changes from z_0 to z_1 the integrands singularities move (middle), crossing the path c_{ab}, however the path can be deformed to avoid the singularities in their new positions (right). This path deformation means that the integral, $f(z)$ is analytic, even though the integrand, $g(z, \alpha)$, is singular.

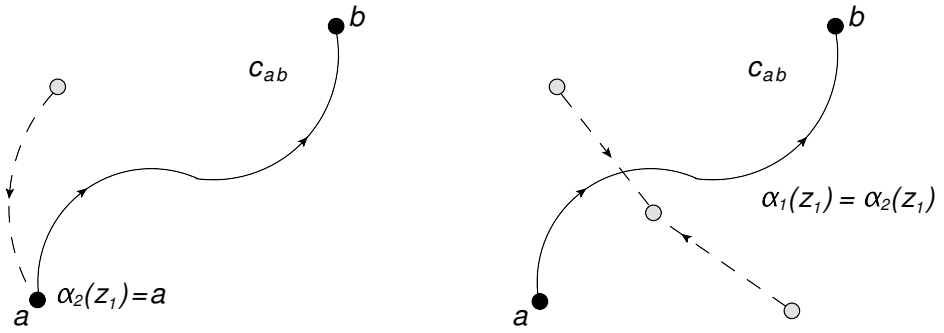


Figure 7.2: From left to right: an end point singularity, in which the singularity and the end point coincide, and a pinch singularity, in which two singularities coincide. The singularity is depicted by the grey dot, labelled by α. Its movement is shown by the dashed line. The end points of the integration path are shown by solid dots and are labelled by a and b.

singularities in Chapter 4. Instances which render analytic continuation impossible, so that the integral $f(z)$ is not analytic on a given domain $\Xi \subset \mathbb{C}$ include:

1. If a singularity $\alpha_i(z)$ of the integrand $g(z, \alpha)$ approaches an end point a or b, such that $\alpha_i(z_1) = a$, termed an *end point* singularity (see Fig. 7.2, left);

2. Two singularities of $g(z, \alpha)$ approaching each other, $\alpha_1(z_1) = \alpha_2(z_2)$, from different directions with respect to the integration path. This is a *pinch* singularity (see Fig. 7.2, right);

3. When avoiding the singularities of the integrand $g(z, \alpha)$ requires a deformation

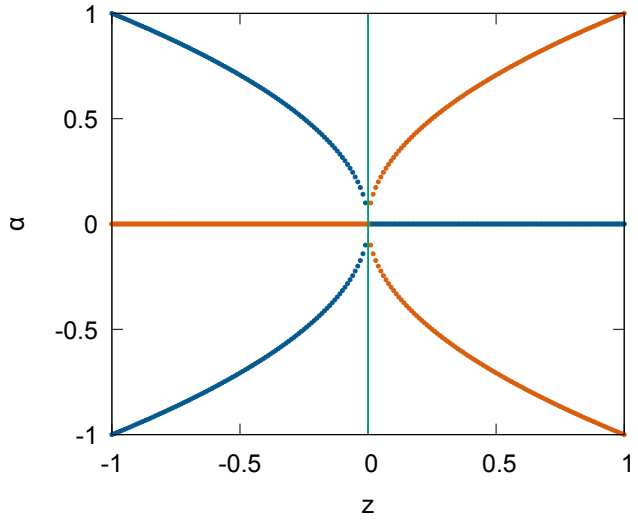


Figure 7.3: The real (blue) and imaginary (orange) parts of the poles of the integrand in Eq. (7.2), $\alpha_1 = i\sqrt{z}$ and $\alpha_2 = -i\sqrt{z}$, which converge on the integration path (green) from -1 to 1 as $z \to 0_+$.

of the path to infinity. As this means that the integration path is trapped by singularities, it also corresponds to a pinch singularity.

Integrals which exhibit such non-analytic behaviour and hence are not *entire* functions, include:

1. $$f(z) = \int_{-1}^{+1} \frac{d\alpha}{\alpha^2 + z} = \frac{2}{\sqrt{z}} \arctan \frac{1}{\sqrt{z}} \ . \tag{7.2}$$

   This integral has a pinch singularity at $z = 0$. This is because, as $z \to 0_+$, the two integrand poles, $\alpha_1 = i\sqrt{z}$ and $\alpha_2 = -i\sqrt{z}$, converge on the integration path, trapping it. This is illustrated in Fig. 7.3. Thus $f(z)$ is non analytic on $\mathbb{C}$.

2. $$f(z) = \int_0^1 \frac{d\alpha}{(\alpha - 2)(\alpha - z)} = \frac{1}{z - 2} \ln \left(\frac{2(z-1)}{z} \right) \ . \tag{7.3}$$

   This integral has end point singularities at $z = 0$ and $z = 1$ and a pinch singularity at $z = 2$, because then the integration path is trapped by the integrand poles at $\alpha = 2$ and $\alpha = z$. Hence $f(z)$ is not analytic on $\mathbb{C}$.

However, within certain sub-domains of $\mathbb{C}$ these integrals $f(z)$ are analytic.

7.3 The Landau Conditions

7.3.1 The Source of Threshold Singularities

The threshold singularities of Feynman intergals are pinch singularities. To demonstrate this requires making the transition to multi-variable integration. Landau and others, [112], [113], extensively discussed the problem of singularities in the space of external variables for multi-variable integration, focussing specifically on Feynman integrals. As previously stated in Chapter 4 and reiterated here for illustrative purposes, a generic Feynman loop integral G in an arbitrary number of dimensions D at loop level L with N propagators, wherein the propagators $P_{\tilde{j}}$ with mass $m_{\tilde{j}}$ can be raised to arbitrary powers $\nu_{\tilde{j}}$, has the momentum space representation

$$G^{\mu_1\cdots\mu_R}_{\alpha_1\ldots\alpha_R}(\{p\},\{m\}) = \left(\prod_{\alpha=1}^{L}\int \mathrm{d}^D\kappa_\alpha\right)\frac{k^{\mu_1}_{\alpha_1}\cdots k^{\mu_R}_{\alpha_R}}{\prod_{\tilde{j}=1}^{N} P_{\tilde{j}}^{\nu_{\tilde{j}}}(\{k\},\{p\},m^2_{\tilde{j}})}\,, \tag{7.4}$$

$$\text{with } \mathrm{d}^D\kappa_\alpha = \frac{\mu^{4-D}}{i\pi^{\frac{D}{2}}}\,\mathrm{d}^Dk_\alpha\,,\ P_{\tilde{j}}(\{k\},\{p\},m^2_{\tilde{j}}) = q^2_{\tilde{j}} - m^2_{\tilde{j}} + i\delta\,, \tag{7.5}$$

where the $q_{\tilde{j}}$ are linear combinations of external momenta p_i and loop momenta k_α. After introducing Feynman parameters, Eq. (7.4) can be rewritten as follows:

$$G^{\mu_1\cdots\mu_R}_{\alpha_1\ldots\alpha_R}(\{p\},\{m\}) = \int \mathcal{D}k \int_0^1 \mathcal{D}t \frac{1}{(F+i\delta)^\nu}\,, \tag{7.6}$$

wherein;

$$\nu = \sum_{\tilde{j}=1}^{N}\nu_{\tilde{j}}\,, \tag{7.7}$$

$$\mathcal{D}k = \left(\prod_{\alpha=1}^{L}\int \mathrm{d}^D\kappa_\alpha\right)\,, \tag{7.8}$$

$$\mathcal{D}t = \prod_{\tilde{j}=1}^{N} dt_{\tilde{j}}\,\delta\left(1-\sum_{\tilde{j}=1}^{N} t_{\tilde{j}}\right)\,, \tag{7.9}$$

$$F = \sum_{\tilde{j}=1}^{N} t_{\tilde{j}}\left(q^2_{\tilde{j}} - m^2_{\tilde{j}}\right)\,. \tag{7.10}$$

It is F which is of paramount importance to understanding the analyticity of Feynman integrals, as it contains integrand singularities ($F = 0$) in the $t_{\tilde{j}}$ which depend on the external variables p_i, the external momenta in this case, through their dependence on $q_{\tilde{j}}$.

7.3.2 Parametrising the Threshold Singularities

It is not sufficient to express the singularities in terms of the $q_{\tilde{j}}$ as they contain loop momenta which are integration variables. Instead, the singular behaviour must be ex-

pressed in terms of the external scales in the problem, p_i, via their dependence on scalar products of the $q_{\tilde{j}}$. Thus, the singular scalar products in the $q_{\tilde{j}}$ must be found, $q_{\tilde{j}_1} \cdot q_{\tilde{j}_2}$. To locate the scalar products of $q_{\tilde{j}}$ which give rise to integrand singularities, in the multi-variable case, is highly mathematically intricate, however the formalism is deceptively simple to state. The singularities of the integrand arise at the configurations given by the Landau conditions, which read:

$$(i)\ q_{\tilde{j}}^2 = m_{\tilde{j}}^2 \text{ or } t_{\tilde{j}} = 0,\ \forall \tilde{j}\,, \tag{7.11}$$

$$(ii)\ \sum_{\tilde{j} \in loop(l)} t_{\tilde{j}} (q_{\tilde{j}})^\mu = 0,\ l = 1...L\ . \tag{7.12}$$

The first condition gives rise to singularities because each summand in F is zero. The case of $q_{\tilde{j}}^2 = m_{\tilde{j}}^2$ can be interpreted as each propagator going on-shell and contributing to the singularity. On the other hand, $t_{\tilde{j}} = 0$ means that the $\tilde{j}^{\text{th}}$ propagator does *not* enter the singularity. The first Landau condition allows the $q_{\tilde{j}_1} \cdot q_{\tilde{j}_2}$ to be found in the case of $\tilde{j}_1 = \tilde{j}_2$. The second Landau condition can be explained geometrically. It means that the singularity surfaces (in the $t_{\tilde{j}}$) are all parallel to each other. So, the hyperpath of Feynman integration cannot be smoothly deformed away to avoid crossing the singularity surfaces when they move, leading to a multi-variable pinch singularity. The second Landau condition results in a set of equations which can be solved to find the $q_{\tilde{j}_1} \cdot q_{\tilde{j}_2}$ inner products where $\tilde{j}_1 \neq \tilde{j}_2$, however it is very difficult to solve in general. To make use of the second Landau condition, it is convenient to contract it with $(q_{\tilde{j}_2})_\mu$, to give $(q_{\tilde{j}_1})^\mu (q_{\tilde{j}_2})_\mu t_{\tilde{j}_2} = 0$ which yields a matrix equation,

$$Q\vec{t} = \vec{0}. \tag{7.13}$$

This is helpful because for non-vanishing $\vec{t}$ the solution of the second Landau condition can be obtained by solving $\det Q = 0$.

7.3.3 Classifying the Thresholds

The solutions of the Landau conditions can be classified according to their location in Feynman parameter space, as follows:

1. Real Thresholds: Those solutions which are within the region of Feynman paramater integration, $0 < t_{\tilde{j}} < 1$. Feynman parameters and kinematic scales corresponding to real thresholds are denoted as $t_{\tilde{j}}^+$, p_i^+ respectively.

2. Pseudo Thresholds: Those solutions which are *not* within the region of Feynman paramater integration, $t_{\tilde{j}} < 0$, $t_{\tilde{j}} > 1$. Feynman parameters and kinematic scales corresponding to pseudo thresholds are denoted as $t_{\tilde{j}}^-$, p_i^- respectively.

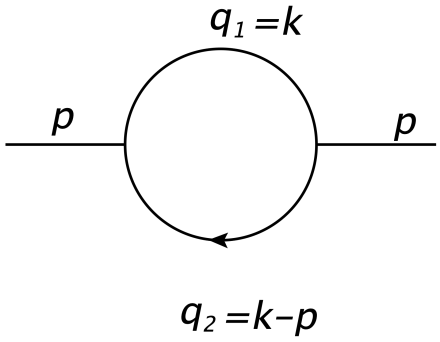


Figure 7.4: A bubble diagram with loop momentum k and external momentum p. The $q_{\tilde{j}}$ read $q_1 = k^2 - m_1^2$ and $q_2 = (k-p)^2 - m_2^2$.

3. Second Type Singularity: these singularities, first noted by Cutkosky [113], are those external momentum configurations at which $\det Q(m_{\tilde{j}} = 0) = 0$. They are denoted as p_i^0.

4. Anomalous Thresholds: for Feynman integrals with several propagators (and hence several possible on-shell configurations) there is another class of thresholds that may reside within the region of integration, *anomalous thresholds* [114]. Such thresholds are characterised by all of the propagators of a Feynman integral contributing to an on-shell configuration (termed the leading Landau singularity). Thus, at an *anomalous threshold*, the singularities in the kinematic scales, p_i^2 are all interdependent. Because of this interdependence, whether or not these thresholds are within the integration region for a given kinematic scale, p_{i_1}, *depends* on the remaining p_i^2. Historically, the anomalous threshold first appeared in the research conducted into electromagnetic form factors (which are connected to three-point functions), because it was present in the hyperon form factor but not in that of the pion or kaon [115].

This classification is summarised in Table 7.1.

Table 7.1: The different types of threshold and the conditions under which they sit in the region of Feynman integration.

Threshold	Real	Pseudo	Second type	Anomalous
Region of integration?	yes	no	no	p_i^2-dependent

7.4 Solving the Landau Conditions for a Bubble Integral

It is not always possible to solve the Landau equations described above, so to illustrate the process the bubble diagram in Figure 7.4 is taken as an example. To locate the

thresholds of this diagram, the following steps are taken:

1: Identify the scales in the problem and find the $q_{\tilde{j}}$

1. There is one external momentum, p and one loop momentum k. So, in this case the denominator F (Eq. (7.6)) reads $t_1(k^2-m_1^2)t_2((k-q)^2-m_2^2)$, thus $q_1 = k$ and $q_2 = (k-p)$. There is only one external scale, p, which by momentum conservation is equal to $p^2 = (q_1+q_2)^2$. Thus the q_1^2, q_2^2 and $q_1 \cdot q_2$ at which the integral is singular need to be found using the equations that make up the two Landau conditions. This will yield the singular positions of the Feynman integrand *in terms of* the kinematic scales, p, m_1 and m_2.

Outcome (1): $q_1 = k$ and $q_2 = (k-p)$

2: Elaborate upon the first Landau condition

2. Having identified the $q_{\tilde{j}}$, the first Landau condition, Eq. (7.11), now reads $q_1^2 = k^2 = m_1^2$ and $q_2^2 = (k-p)^2 = m_2^2$ for non-vanishing $t_{\tilde{j}}$. These specify the positions at which the denominator of the Feynman integral, F, becomes zero and the propagators are on-shell. However, to find the value of p at which the Feynman integrand is singular $q_1 \cdot q_2$ is still needed.

Outcome (2): $q_1^2 = k^2 = m_1^2$ and $q_2^2 = (k-p)^2 = m_2^2$

3: Solve det$Q = 0$

3. $Q_{\tilde{j}_1\tilde{j}_2} = q_{\tilde{j}_1} \cdot q_{\tilde{j}_2}$ thus the matrix elements of Q are $Q_{11} = q_1^2 = m_1^2$, $Q_{12} = q_1 \cdot q_2$, $Q_{21} = q_2 \cdot q_1$ and $Q_{22} = q_2^2 = m_2^2$, using the first Landau condition. Inserting these into det$Q = 0$ yields

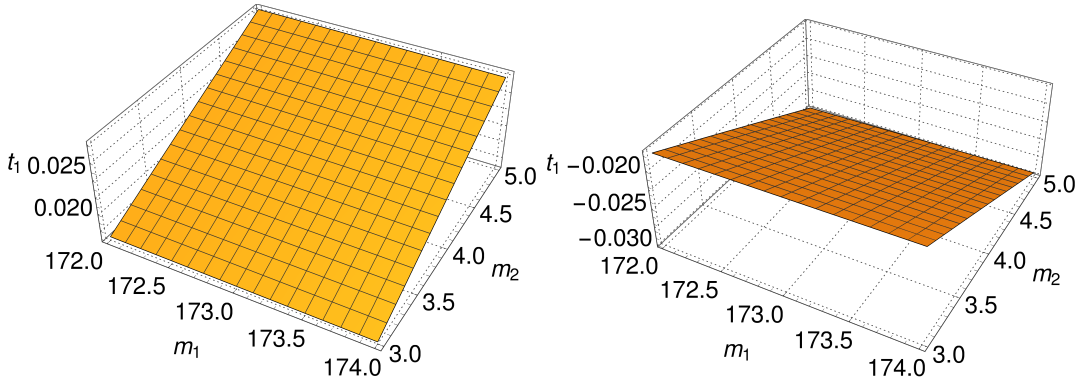
$$\det \begin{bmatrix} m_1^2 & q_1 \cdot q_2 \\ q_2 \cdot q_1 & m_2^2 \end{bmatrix} = 0\,, \tag{7.14}$$

which has the solution:

$$q_1 \cdot q_2 = \pm m_1 m_2\ . \tag{7.15}$$

This provides the missing piece of information.

Outcome (3): $q_1 \cdot q_2 = \pm m_1 m_2$



(a) (b)

Figure 7.5: The values of t_1^+, (a) and t_1^-,(b) respectively, for $m_1 \in [172, 174]$ GeV^2 and $m_2 \in [3, 5]$ GeV^2. This depicts the fact that the threshold $p_{(+)}$ lies within the region of integration but the pseudo threshold $p_{(-)}$ does not.

4: Express the singularities in terms of the external scales in the Feynman integral, p_i

4. Having utilised the equations in the Landau conditions, the scalar products in the $q_{\tilde{j}}$ that are fulfilled when the Feynman integral is singular are $q_1^2 = m_1^2$, $q_2^2 = m_2^2$ and $q_1 \cdot q_2 = \pm m_1 m_2$. Thus the thresholds of the bubble Feynman integral are positioned at $p^2 = (m_1 \pm m_2)^2$. Explicitly, the two thresholds are

$$\begin{bmatrix} p_{(+)}^2 \\ p_{(-)}^2 \end{bmatrix} = \begin{bmatrix} (m_1 + m_2)^2 \\ (m_1 - m_2)^2 \end{bmatrix} . \tag{7.16}$$

From this it is clear that the threshold singularity corresponds to the introduction of a new pair of particles, with masses m_1 and m_2.

Outcome (4): $p^2 = (m_1 \pm m_2)^2$

5: Distinguish Between Real Thresholds and Pseudo Thresholds

5. To understand which of the singularities given by the Landau conditions actually lie within the region of integration it is necessary to solve $Q\vec{t} = \vec{0}$ for $\vec{t}$, the vector of Feynman parameters. For the bubble integral, this is given by

$$\vec{t} = \begin{bmatrix} t_1 \\ (1 - t_1) \end{bmatrix} , \tag{7.17}$$

due to the Dirac delta function $\delta\left(1-\sum_i^N t_i\right) = \delta\left(1-t_1-t_2\right)$ that arises when Feynman parameterisation is performed. Thus, $Q\vec{t}=\vec{0}$ reads:

$$\begin{bmatrix} m_1^2 & q_1\cdot q_2 \\ q_2\cdot q_1 & m_2^2 \end{bmatrix}\begin{bmatrix} t_1 \\ (1-t_1) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}, \tag{7.18}$$

which leads to the system of equations

$$\begin{aligned} &m_1^2 + q_1\cdot q_2(1-t_1) = 0\,, \\ &(q_1\cdot q_2)t_1 + (1-t_1)m_2^2 = 0\ . \end{aligned}$$

Substituting in $q_1\cdot q_2 = \pm m_1 m_2$ from the second Landau condition (Eq. (7.15)) yields $t_1(\pm m_1 m_2 - m_1^2) + m_1 m_2 = 0$ which reveals the position in Feynman parameter space at which the thresholds in Eq. (7.16) sit, namely

$$t_1 = \frac{m_1 m_2}{-m_1^2 \pm m_1 m_2} = \frac{m_2}{m_2 \pm m_1}\,, \tag{7.19}$$

labelled as

$$t_1^{\pm} = \frac{m_2}{m_2 \pm m_1}\,, \tag{7.20}$$

which illuminates the Feynman parameter space allowing the thresholds to be located. The masses are positive, so $m_1, m_2 > 0$ and thus the first threshold is located at $0 < t_1^+ < 1$ but for the second $t_1^- > 1$ or $t_1^- < 0$. Thus the first threshold, p_+^2, is within the region of integration. However the second threshold, p_-^2 corresponds to a Feynman parameter which is *not* within the region of integration and so the integration path would have to be deformed well outside the relevant region to make contact with it. The threshold p_-^2 is referred to as a pseudo threshold.

Outcome (5): p_-^2 is a pseudo threshold

7.5 The Importance of Real and Pseudo Thresholds for TayInt

The distinction between real and pseudo thresholds is of crucial importance when it comes to evaluating Feynman integrals. As discussed in the introduction, the objective of the TayInt program is to produce a result for a Feynman integral as a function of the kinematic scales which can be evaluated quickly to an accurate and precise numerical approximation regardless of the region of phase space. But the presence of threshold singularities, dependent on the kinematic scales, which sit within the region of integration in the Feynman parameter space are problematic for the following reasons:

1. The Feynman integrand cannot be Taylor expanded along the real integration path because of the singularity, thus an algebraic result which converges to an accurate, precise numerical approximation cannot be produced in this kinematic region,

2. The position of the singularity depends on the kinematics, the very thing that must be contained in the final result, so it is not possible to simply subtract the singularity.

The contour selection part of TayInt was written to produce algebraic approximations which evaluate to precise and accurate numerical approximations in over-threshold regions. This finds the correct contour of integration that avoids the thresholds in each region, leading to a library of algebraic approximations, one for each threshold region. As they do not sit within the region of integration, pseudo thresholds do not present a problem for the production of algebraic approximations for Feynman integrals with TayInt but the method of threshold location TayInt uses is not based upon the Landau conditions and cannot distinguish between real and pesudo-thresholds, thus the two are treated equally. Whilst this means that the final TayInt algorithm takes slightly longer to run than is ideal, it also ensures that the systematic algebraic approximations converge to accurate, precise numerical values in all kinematic regions.

7.6 Classifying the Solutions of the Landau Conditions for a Bubble Integral

Revisiting the bubble integral (Fig. 7.4), the first Landau condition gives the equations

$$(k-p)^2 = m_1^2,\ k^2 = m_2^2\,, \tag{7.21}$$

corresponding to two hyperboloids with a relative central displacement of p. The second Landau condition leads to the equation:

$$t_1 k_\mu + (1-t_1)(k-p)_\mu = 0\,, \tag{7.22}$$

which means that $k_\mu = (1-t_1)p_\mu$ and so guarantees that k and p are parallel in momentum space.

7.6.1 Real Thresholds, Pseudo Thresholds and Second Type Singularities

Taking $k = (k_0, 0, 0, k_3)$, Eqs. (7.21) and (7.22) admit the solutions:

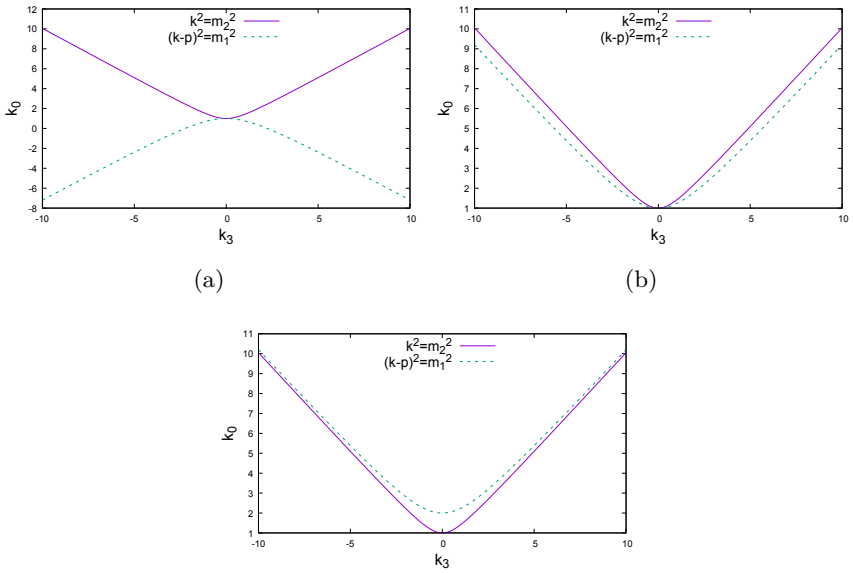
Real Threshold: $\boldsymbol{p = p_+ = (m_1 + m_2, 0, 0, 0), t_1 \in [0, 1]}$

1.

$$k^2 = m_2^2 \to k_0 = \pm\sqrt{k_3^2 + m_2^2}\,, \tag{7.23}$$

and

$$(k-p)^2 = m_1^2 \to k_0 = (m_1 + m_2) \pm \sqrt{k_3^2 + m_1^2}\,, \tag{7.24}$$



(a) (b)

(c)

Figure 7.6: The solutions of the Landau equations for the bubble integral, evaluated at; (a) the real threshold, $p = p_+ = (m_1 + m_2)$, (b) the pseudo threshold $p = p_- = (m_1 - m_2)$, (c) the second type singularity. In all cases, $m_1 = 1$, $m_2 = 2$ and $k = (k_0, 0, 0, k_3)$. The dashed lines show the solutions to $(k-p)^2 = m_2^2$.

Pseudo Threshold: $p = p_- = (m_1 - m_2, 0, 0, 0), t_1 \notin [0, 1]$

2.

$$k^2 = m_2^2 \to k_0 = \pm\sqrt{k_3^2 + m_2^2}\,, \tag{7.25}$$

and

$$(k-p)^2 = m_1^2 \to k_0 = (m_1 - m_2) \pm \sqrt{k_3^2 + m_1^2}\,, \tag{7.26}$$

Second type singularity $p = p^0 = (0, 0, 0, 0)$

3.

$$k^2 = m_2^2 \to k_0 = \pm\sqrt{k_3^2 + m_2^2}\,, \tag{7.27}$$

and

$$(k-p)^2 = m_1^2 \to k_0 = \pm\sqrt{k_3^2 + m_1^2}\,. \tag{7.28}$$

The momentum-space solutions for each type of Landau singularity are shown in Fig. 7.6, with (a) and (b) depicting the real and pseudo thresholds respectively, as well as the

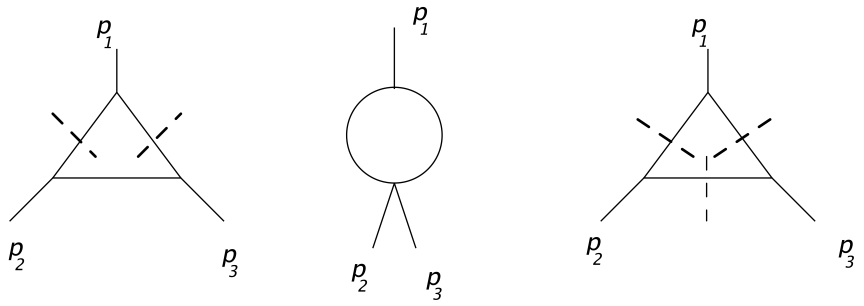


Figure 7.7: From left to right: a one-loop triangle with cuts that lead to a real threshold (sub-leading), the reduced diagram with $t_1 = 0$ in the Landau equations that contains the sub-leading thresholds, the one-loop triangle with the cuts that correspond to an anomalous threshold.

second type singularity. As previously stated in Subsection 7.3.3, these singularities satisfy $\det Q(m_{\tilde{j}} = 0) = 0$, where the Q is the matrix of inner products $q_{\tilde{j}_1} \cdot q_{\tilde{j}_2}$. In the case of the bubble this singularity is easily obtained as when $p^2 = 0$ and hence the second type singularity is $p^0 = 0$. Using the second Landau condition (Eq. (7.25)), the value of the Feynman parameter t_1 at which this singularity is located can be found, as follows:

$$t_1 k_\mu + (1 - t_1)(k - p)_\mu = 0 \tag{7.29}$$

$$\rightarrow t_1(k \cdot p) + (1 - t_1)(k \cdot p - p^2) = 0 \tag{7.30}$$

$$\rightarrow t_1 = \frac{p^2 - k \cdot p}{p^2}\,, \tag{7.31}$$

so as $p \to 0$, $t_1 \to \infty$. So, the second type singularity is not in the region of integration, as must be the case due to its mass independence (looking back at the analysis in and around Eq. (7.18)).

7.6.2 Anomalous Thresholds

For the bubble integral, only the real threshold is present within the integration region but for Feynman integrals with more propagators multiple thresholds can be located there. For example, the triangle diagram in Fig. 7.7 corresponds to a Feynman integral with three propagators and so there are more than two on-shell configurations of the propagators (which can be visualised by noting that the diagram can be sliced into more than two pieces). If all three of the propagators enter an on-shell configuration then this corresponds to an *anomalous threshold*, in which the singularities in the kinematic scales, p_1^2, p_2^2, p_3^2 are all interdependent, unlike in the case of the bubble when the threshold positions only depended on the masses (Eq. (7.11)).

Thus, expressing the thresholds in terms of p_1^2, whether or not these thresholds are within the integration region *depends* on p_2^2 and p_3^2. An example of an anomalous threshold residing within the integration region is provided by the one-loop triangle in Fig. 7.7.

In the case when $p^2 = p_2^2 = p_3^2$ and $m = m_2 = m_3$, the external scales are p^2 and p_1^2 and the external masses are m^2 and m_1^2. This kinematic constellation leads to an anomalous threshold when $p_1^2 \geq 4m^2 - (p^2 - (m_1^2 + m_1^2))^2/m_1^2$, provided $p^2 > m^2 + m_1^2$ [115]. This threshold is anomalous in the sense that it is actually lower than $4m^2$, at which the two-propagator threshold sits in the reduced diagram with $t_1 = 0$ (Fig. 7.7, middle).

7.7 Using and Interpreting the Landau Conditions

7.7.1 Cutkosky Cutting Rules

Theory

The Landau conditions specify the locations of the thresholds of Feynman integrals but the Cutkosky cutting rules [113] give the actual singularities. In more detail, the Landau equations state that there is a singularity if the conditions in Eqs. (7.11) and (7.12) are met, the Cutkosky rules say that the singularity itself can be found by making the following replacement:

$$\frac{1}{q_{\tilde{j}}^2 - m_{\tilde{j}}^2 + i\delta} \to 2\pi i \delta^{(+)}\left(q_{\tilde{j}}^2 - m_{\tilde{j}}^2\right) \tag{7.32}$$

where $\delta^{(+)}\left(p^2 - m^2\right) = \delta\left(p^2 - m^2\right)\theta\left(p_0\right)$.

Thus, the Feynman integral G, given in Eq. (7.4)
has a discontinuity cut of:

$$\text{Disc}[G] = (-2\pi i)^r \int \mathcal{D}k \frac{\prod_{i=1}^{r} \delta^{(+)}(q_i^2 - m_i^2)}{\prod_{j=1}^{N-r}\left(q_{r+j}^2 - m_{r+j}^2\right)}, \tag{7.33}$$

for$i \in [1, ..., r]$ with $r \leq N$, the number of propagators
and momenta assumed real .

Demonstration for the Bubble Integral

This is demonstrated for the bubble integral in Peskin and Schroeder [93] and works for any Feynman integral. A proof is given in the original paper [113]. As an outline, if an integral has a pole $\alpha_1(z)$ which tends to m_1^2 for some value of z (where z is a function of external and internal momenta), trapping the integration path with a pinch singularity, then the integration path is deformed below m_1^2, so that it encloses the singularity. The integral is then carried out using Cauchy's theorem which gives rise to the $\delta(q_i^2 - m_i^2)$. The $\theta(q_0)$ enforces the fact that the singularity only lies on the integration path in the physical region.

7.7.2 The Coleman-Norton Interpretation

Theory

The actual physical interpretation of the second Landau condition, Eq. (7.12), was provided by Coleman and Norton, [116], who related the Feynman parameter and loop momenta products with four-vectors describing the propagation of a particle by making the identification

$$t_{\tilde{j}} k^{\mu}_{\tilde{j}} = \Delta x^{\mu}_{\tilde{j}} \,, \tag{7.34}$$

which allows $t_{\tilde{j}}$ to be written as $t_{\tilde{j}} = \frac{\Delta x^0_{\tilde{j}}}{k^0_{\tilde{j}}}$. Using this,

$$\Delta x^{\mu}_{\tilde{j}} = t_{\tilde{j}} k^{\mu}_{\tilde{j}} = \frac{\Delta x^0_{\tilde{j}}}{k^0_{\tilde{j}}} k^{\mu}_{\tilde{j}} = \Delta x^0_{\tilde{j}} \left(1, \frac{\vec{k}_{\tilde{j}}}{k^0_{\tilde{j}}} \right) \,, \tag{7.35}$$

which illustrates that $\Delta x^{\mu}_{\tilde{j}}$ is a four-vector, the mathematical representation of a classical on-shell particle propagating through space-time with momentum k.

Demonstration for the One-Loop Triangle

Now take the one-loop triangle, as shown on the left of Figure 7.7. For this diagram, the propagators are $k^2 - m^2$, $(k - p_1)^2 - m^2$, $(k + p_2)^2$ assuming equal masses and that all external momenta are incoming. Thus $q_1 = k$, $q_2 = (k - p_1)$, $q_3 = (k + p_2)$ and so the second Landau condition, Eq. (7.12), reads:

$$\sum_{\tilde{j} \in loop(l)} t_{\tilde{j}} (q_{\tilde{j}})^{\mu} = 0 \implies t_1 k^{\mu} + t_2 (k - p_1)^{\mu} + t_3 (k + p_2)^{\mu} = 0 \,, \tag{7.36}$$

The one-loop triangle has seven reduced diagrams, three of which come from pinching one line, three from pinching two lines and one from pinching all three lines. These are shown on the first, second and third rows of Fig. 7.8 respectively. Considering these reduced diagrams, it is clear that only the leftmost two on the first row and the diagram on the third row correspond to the propagation of a classical particle, as classical particles do not propagate via external closed loops. The first diagram in Fig. 7.8 contains a sub-loop,

$$\int d^4 k \frac{1}{k^2 - m^2} \cdot \frac{1}{(p_1 - k)^2 - m^2} \,, \tag{7.37}$$

with loop momentum k and external momentum p_1, which, written as a superposition of four-vectors, reads $\Delta x^{\mu}_{p_1 - k} - \Delta x^{\mu}_{k} = 0$. Note that this is simply the bubble diagram considered in Fig. 7.2. Using Eq. (7.34) to replace $\Delta x^{\mu}_{\tilde{j}}$ leads to

$$t_2 (p_1 - k)^{\mu} - t_1 k^{\mu} = 0. \tag{7.38}$$

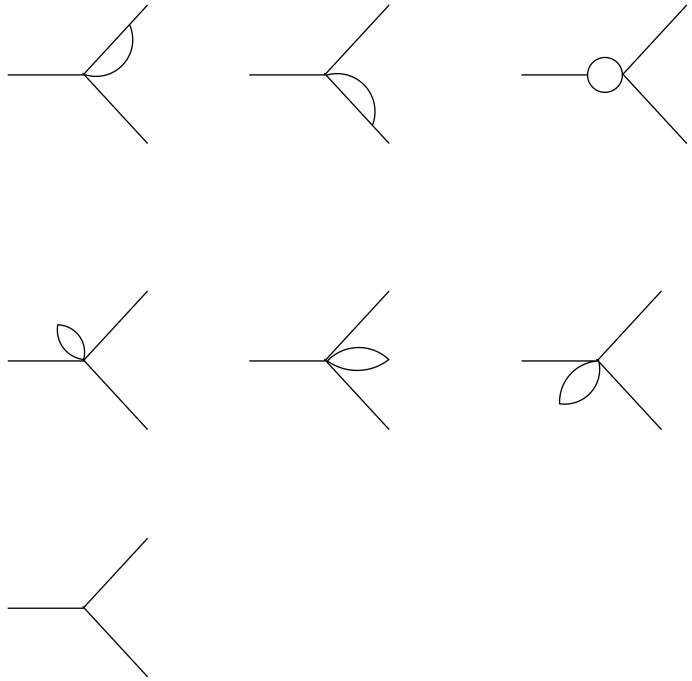


Figure 7.8: The reduced diagrams that can be generated by pinching one, two and three lines respectively out of the one-loop triangle on the left of Fig. 7.7.

This is the second Landau equation for the one-loop triangle, Equation(7.36), with $t_3 = 0$. The second diagram in Fig. 7.8 generates the second Landau equation with $t_2 = 0$. The final reduced diagram has no loops and so corresponds to $k^\mu = t_1 = t_2 = t_3 = 0$, which is the soft solution of the second Landau equation.

Formal Statement

This leads us to the Coleman-Norton interpretation:

Classical reduced diagrams give solutions to the Landau equations.

7.8 Summary

Thus, the singularities of Feynman integrals in the space of external kinematics are located using the Landau conditions (Eqs. (7.11)-(**??**)). These singularities can be classified according to their location in the space of Feynman parameters, but in general the solution of the Landau equations is still the subject of ongoing research. Now that the concept of threshold singularities has been formally introduced, in the next chapter, the challenge they pose to the TayInt approach will be discussed.

8 | The Minkowski Problem

8.1 Introduction: Attempting Kinematic Generalisability

As stated in the introduction of this thesis, the work presented here is geared towards producing a library of systematic, kinematically algebraic approximations for Feynman integrals at the level currently obstructing further progress in the prediction of differential cross sections in theoretical particle physics, namely, the two-loop, four-point level at which elliptic structures appear. To be of use in contemporary research, these algebraic approximations must be able to be *quickly* evaluated at *arbitrary* kinematic points of physical interest and yield *accurate* and *precise* values for the starting Feynman integral. The scope of this problem has been provided in the first half of this thesis, which should leave no doubt of the truth of the claim with which it opened, that, in general, calculating Feynman integrals at the two-loop level is difficult. However, the *second* half of this thesis is devoted to presenting a new and novel algorithm for tackling this difficult calculation, by pursuing a Taylor expansion in the Feynman parameters such that the approximation is algebraic in the kinematic scales of the integral. In this chapter, the particular difficulty of attempting to calculate Feynman integrals in this way will be introduced. Firstly, by illustrating the consequences of attempting a plain Taylor expansion of the subsectors of a two-loop, three-point Feynman integral and secondly, by numerically evaluating it in the Minkowksi kinematic region. The idea that forms the nucleus of the final TayInt algorithm will then be revealed.

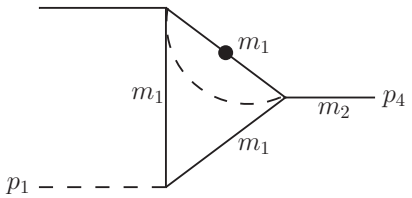


Figure 8.1: The two-loop triangle I10 for which analytical results are available [92].

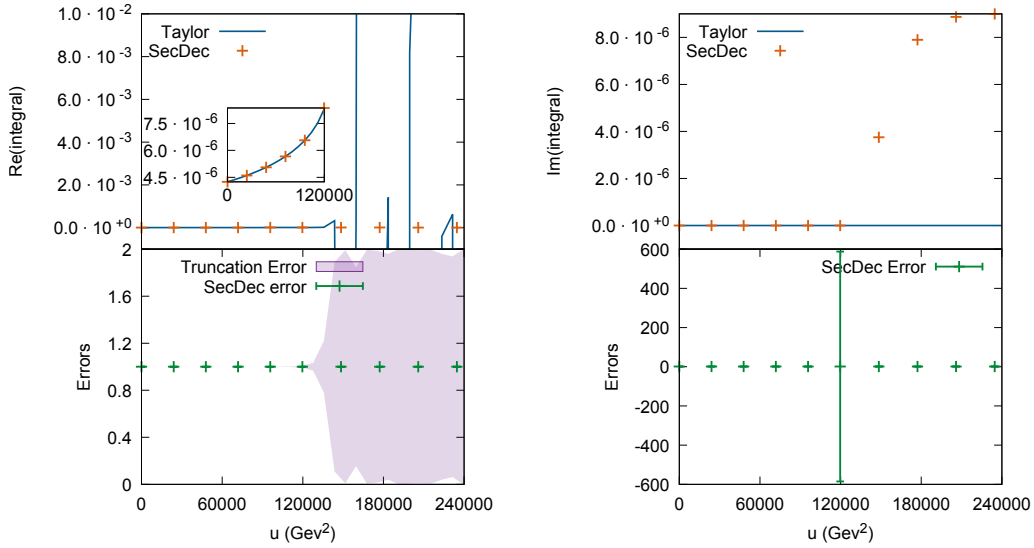


Figure 8.2: The integrated plain Taylor expansion and numerical SecDec results for subsector 1 of I10 at order ϵ^0 with $u \in [0m_1^2, 8m_1^2]$ and $m_h^2 = 0.5m_1^2$, $m_1 = m_t = 173$ GeV. The left panels show the real parts and the right hand panels the imaginary parts. The lower panels display the associated errors and the inset plot depicts the behaviour of the approximation with $u \in [0, 4m_1^2]$. As the plain Taylor expansion has no imaginary part, the truncation error associated with the imaginary part is undefined. This demonstrates the effect of the thresholds on approximating a Feynman integral by means of a Taylor expansion.

8.2 Plain Taylor Expansion and Integration of the I10 Integral

Once a finite Feynman integral has been decomposed into subsectors, it is free of any singularities due to the Feynman parameters within its region of integration. This seems to suggest that these subsectors can be normally Taylor expanded and integrated in the Feynman parameters to generate approximations algebraic in the kinematic scales. If this is carried out for subsector 1 of the integral I10, Fig. 8.1, at order ϵ^1 and the following numerical values inserted: $u \in [0, 8m_1^2]$ and $m_h^2 = 0.5m_1^2$, $m_1 = m_t = 173$ GeV, then the subsequent approximation behaves as shown in Fig. 8.2. Upon viewing this Figure, it becomes apparent that the numerical approximations for subsector 1 of the integral I10 at order ϵ^0 obtained via a plain Taylor expansion exhibit two distinct regions. The first such region is characterised by the following properties:

1. The numerical approximation for the subsector is fully real.

2. The approximation obtained via a Taylor expansion is valid and well-converging.

The second exhibits by the following features:

1. The numerical approximation for the subsector is complex.
2. The approximation obtained via a Taylor expansion breaks down and is poorly-converging.
3. At the threshold between regions, the numerical SecDec result also breaks down.

8.3 The Over-Threshold Problem

The aim of this project is to produce a library of algebraic approximations for Feynman integrals which can be quickly evaluated to yield precise and accurate results irrespective of the kinematic points inserted. Therefore, algebraic approximations which break down in the kinematic region of greatest phenomenological interest is quite unacceptable.

From Fig. 8.2 it is clear that, although it turns the two-loop Feynman integrand in question into a polynomial which can easily be integrated, simply Taylor expanding the subsectors of the Feynman integral will not lead to algebraic approximations which yield accurate results at all kinematic points. This is because the approximations lose accuracy and precision once kinematic points exceeding the threshold of the integral are inserted. The I10 integral is the simplest integral that will be considered in this thesis, having the fewest Feynman parameters, subsectors and external legs. The fact that the threshold singularities within I10 already lead to the uncontrolled growth of the coefficients of a plan Taylor expansion adds detail to what was stated in the introduction: that in general two-loop Feynman integrands are not well suited to a plain Taylor expansion. This is due to the presence of threshold singularities which appear in the Minkowski region.

8.4 The Diagnosis

Before deciding how to turn a two-loop Feynman integrand into an object that *is* well suited to a Taylor expansion and making progress towards producing a library of algebraic approximations that can be evaluated to yield accurate and precise numerical approximations in a timely manner, the reason for the failure of the plain Taylor expansion must be specified. The fact that, in addition to the approximation obtained by means of a Taylor expansion, the numerical SecDec approximation *also* breaks down at a particular kinematic point indicates that there must be a threshold singularity at that point. As the coefficients of the Taylor expansion are functions of the kinematic scales of the Feynman integral, the presence of a singularity dependent on these variables destroys the convergence of the Taylor expansion. Moreover, the fact that the numerical SecDec approximation for the Feynman integral becomes complex over threshold indicates that a Taylor expansion of the subsectors (which are fully real) is not an approach suitable for achieving the aims of this project. Thus, the method for curing the damage caused

by the threshold singularities and the production of the imaginary part of the Feynman integral are linked and so in order to proceed, the objectives are:

1. To find a way of approximating Feynman integrals which is valid even in the presence of threshold singularities,

2. To introduce an imaginary part into the approximation in the over-threshold regions.

The strategies considered for achieving these objectives are:

1. Remove the threshold singularities: the rationale is to split the integrand into a finite and singular (containing the threshold singularities located within the region of integration) part. The former can be Taylor expanded and integrated, the latter can be integrated exactly using the Residue Theorem. The exact integration of the singular pieces generate the imaginary part of the approximation for the Feynman integral.

2. Avoid the threshold singularities: the rationale is to find a contour of integration configured in the complex space such that no threshold singularities are crossed. Hence, the representation of the integrand using this contour has a well-converging Taylor expansion. The use of a complex contour generates the imaginary part of the approximation for the Feynman integral that arises after the threshold is crossed.

Efforts towards implementing them for subsector 1 of the I10 integral at order ϵ^0 will be presented in the forthcoming.

8.5 Removing Threshold Singularities

Before applying the strategy of removing the threshold singularities of a two-loop integrand to that of subsector 1 of I10, the strategy itself must first be outlined. It consists of three steps:

1. Finding the threshold singularities,

2. Subtracting the threshold singularities,

3. Integrating the remainder and subtraction terms.

8.5.1 Finding the Threshold Singularities

In the first step, the positions of the threshold singularities in the Feynman parameter space of each subsector of the Feynman integral in question are located. This can be done quickly and easily by using the `Solve` command within the MATHEMATICA program. The Feynman parameters in which there are threshold singularities are denoted by t_s and those free of singularities denoted by t_f. The threshold singularity is labelled using the

notation $\tilde{t}_s$, which is a function of the external momenta p_i^2, internal masses m_j^2 and remaining Feynman parameters t_f, as follows:

$$\tilde{t}_s = f(p_i^2, m_j^2, t_f)\,, \tag{8.1}$$

As these threshold singularities present a barricade to the convergence of a Taylor expansion of the subsector integrands, they must next be subtracted in the following step.

8.5.2 Subtracting the Threshold Singularities

Once the threshold singularities within the region of integration have been located, the next step is to subtract them from the subsector integrand, denoted by G_l^F where l runs over the number of subsectors. As the subsector integrands contain singular points at $\tilde{t}_s$, they cannot be Taylor expanded in the Feynman parameters. However, as discussed in Chapter 3, they can be Laurent expanded in each of the t_s about $\tilde{t}_s$,

$$G_l^F(t_s, t_f) = \sum_{n=-\infty}^{\infty} a_n (t_s - \tilde{t}_s)^n\,, \tag{8.2}$$

where

$$a_n = \frac{1}{2\pi i} \oint_{\partial\Xi} \frac{G_l^F(t_s, t_f)}{(t_s - \tilde{t}_s)^{n+1}}\,, \tag{8.3}$$

given Ξ, a domain of integration centred on the singular point, $\tilde{t}_s$. The threshold singularity is of order $n = -1$. Thus, every term in the Laurent series for the $G_l^F(t_s, t_f)$ in t_s is finite, except for the $a_{-1}(t_s - \tilde{t}_s)^{-1}$ term. Therefore, this must be subtracted.

The object a_{-1} is the residue [88] of $G_l^F(t_s, t_f)$ at the point $t_s = \tilde{t}_s$. So, the residue of the subsector integrand at the threshold singularity must be found in order to compute the subtraction term. This can easily be done using the `Residue` command in MATHEMATICA. Performing the subtraction will split the subsectors of the Feynman integral into remainder terms, which can be Taylor expanded and subtraction terms, which must be integrated exactly using the Residue Theorem to generate the imaginary part of the subsector in the over-threshold kinematic region. This will be performed in the following step.

8.5.3 Integrating the Remainder and Subtraction Terms

By construction, the remainder part of each subsector following the subtraction of the threshold singularity is well suited to a Taylor expansion. However, the subsectors G_l^F originally contained threshold singularities within the region of integration, which were stored as singularities in t_s (Eq. (8.1)) and removed from the subsector by means of the subtraction term constructed from its Residue. Thus, this subtraction term must also be integrated. This will be achieved by applying the Residue Theorem to that singularity in t_s. Hence, for this strategy to succeed, the subtraction term must *only* contain the singularity that was removed from the original integrand. The results of both integrals will then be combined to generate algebraic approximations for each subsector, which can in turn be combined to yield such an approximation for the full Feynman integral.

8.6 Application of the Singularity Removal Strategy to the I10 Integral

To test this singularity removal strategy, it is now applied to a two-loop subsector integral, I10 subsector 1 at order ϵ^0, which has the form:

$$\begin{aligned}\mathrm{I10}_1 = \prod_{j=0}^{2} \int_0^1 dt_j \Bigg((1 + t_0 + t_1 + t_0 t_2 + t_1 t_2) \Big[- m_2^2 t_0 - u t_1 + m_1^2 \Big(1 + t_0^2 (1 + t_2) \\ + t_1^2 (1 + t_2) + t_1 (2 + t_2) + t_0 \big[2 + t_2 + t_1 (2 + 2 t_2) \big] \Big) \Big] \Bigg)^{-1} .\end{aligned} \tag{8.4}$$

The first step in removing the threshold singularities of this subsector integral is finding them.

8.6.1 Finding the Threshold Singularities

Using the `Solve` command in MATHEMATICA, it is found that I10 subsector 1 at order ϵ^0 is singular within the region of integration $t_0 \in [0,1]$, $t_1 \in [0,1]$, $t_2 \in [0,1]$. These singularities occur in the t_j at the following points which depend on the kinematic scales:

$$\tilde{t}_2 = \frac{-m_1^2 - 2m_1^2 t_0 + m_h^2 t_0 - m_1^2 t_0^2 - 2m_1^2 t_1 + u t_1 - 2m_1^2 t_0 t_1 - m_1^2 t_1^2}{m_1^2 (t_0 + t_0^2 + t_1 + 2 t_0 t_1 + t_1^2)} , \tag{8.5}$$

so $s = 2$ in this case and $f = 0, 1$. This singularity prevents a Taylor expansion from converging. So, the next step is to subtract it from the integrand, leaving a piece which can be Taylor expanded, in the hope that the resultant subtraction term can be exactly integrated.

8.6.2 Subtracting the Threshold Singularities

Once the threshold singularities within the region of integration have been located, the next step is to subtract them from the integrand of I10 subsector 1, denoted by $\mathrm{I10}_1(t_0, t_1, t_2)$. As the first subsector integrand contains singular points at $t_2 = \tilde{t}_2$, it cannot be Taylor expanded in the Feynman parameters. However, as discussed in the outline of the strategy, it can be Laurent expanded in t_2,

$$\mathrm{I10}_1(t_0, t_1, t_2) = \sum_{n=-\infty}^{\infty} a_n (t_2 - \tilde{t}_2)^n , \tag{8.6}$$

where

$$a_n = \frac{1}{2\pi i} \oint_{\partial \Xi} \frac{\mathrm{I10}_1(t_0, t_1, t_2)}{(t_2 - \tilde{t}_2)^{n+1}} , \tag{8.7}$$

given Ξ, a domain of integration centred on the singular point, $\tilde{t}_2$. As previously outlined in Section 8.5, every term in the Laurent series for $\mathrm{I10}_1(t_0, t_1, t_2)$ in t_2 is finite, except for the $a_{-1}(t_2 - \tilde{t}_2)^{-1}$ term, which must be subtracted.

Computing the Residue of $\mathrm{I10}_1(t_0, t_1, t_2)$ at $\tilde{t}_2$, the subtraction term corresponding to the singularity in Eq. (8.5) therefore reads:

$$\mathrm{Sub}_1^{\mathrm{I10}} = ((t_0 + t_1)(m_h^2 t_0 + u t_1))^{-1} (1 - \tilde{t}_2)^{-1}, \tag{8.8}$$

which, using Eq. (8.5), simplifies to the explicit form:

$$\begin{aligned}\mathrm{Sub}_1^{\mathrm{I10}} =& \frac{m_1^2(1 + t_0 + t_1)}{(t_0 + t_1)(m_h^2 t_0 + u t_1)} \left(m_1^2(1 + t_0 + t_1)(1 + t_0 + t_1 + (t_0 + t_1)t_2)\right. \\ & \left. - (m_h^2 t_0 + u t_1)\right)^{-1},\end{aligned} \tag{8.9}$$

which is subtracted from the subsector integrand. This leaves a remainder that can be Taylor expanded and integrated to yield a well-converging approximation in over-threshold kinematic regions at the end of this step.

8.6.3 Integrating the Remainder and Subtraction Terms

The first subsector of I10 originally contained one threshold singularity within the region of integration. This was stored as a singularity in t_2 (Eq. (8.5)) and removed from the subsector by means of the subtraction term in Eq. (8.9). In the next step, this subtraction term must then be integrated by applying the Residue Theorem to that singularity in t_2. Thus, for this method to work, the subtraction term must only contain the singularity that was removed from the original integrand. However, upon examining the subtraction term, there is not only the singularity in t_2 but another in t_1 that is within the region of integration,

$$\begin{aligned}\tilde{\tilde{t}}_1 = & \frac{-1}{2m_1^2(1 + t_2)} \left(- m_h^2 + m_1^2(2 + t_2 + 2t_1(1 + t_2))\right. \\ & \left. + [m_h^4 + m_1^4 t_2^2 + 4m_1^2 u t_1(1 + t_2) - 2m_1^2 m_h^2(2 + t_2 + 2t1(1 + t_2))]^{\frac{1}{2}})\right)^{-1}.\end{aligned} \tag{8.10}$$

Therefore, to integrate the subtraction term in Eq. (8.9), the singularity $\tilde{\tilde{t}}_1$ must first be subtracted from it, iterating the procedure above. However, doing so would introduce another new singularity in the iterated subtraction term. Because each subtraction generates a new singularity, each subtraction term is a brick in a wall of singularities and so performing the integration of the subsector of I10 in this way would require an infinite number of subtractions. The necessity for iterating the threshold singularity subtraction process is a general feature of two-loop integrands considered in this thesis, of which I10 is the simplest. The fact that the subtraction terms contain more singularities than were originally within the subsector integrand means that the removal strategy is not generally applicable to two-loop integrands. Therefore, it cannot be used to convert them into forms well suited to a Taylor expansion. The second proposed strategy, that of *avoiding* the threshold singularities, is consequently outlined and tested next.

8.7 Avoiding Threshold Singularities

The crux of the problem with the method of removing the threshold singularities is that the contour of integration can only be traversed in one way. This leaves no space to manoeuvre around the singularities, which become more and more packed together in the subtraction terms. To attempt to remedy this, the second strategy for dealing with the threshold singularities applies the following transformation in the Feynman parameters of the subsector, $t_j \rightarrow \frac{1}{2} - \frac{1}{2}e^{-i\theta_j}$. This moves them into the complex domain of integration. Rather than being glued to the real contour of integration, forced to integrate every Feynman parameter from 0 to 1, each θ_j parameter can be integrated along a semi-circle in the clockwise or anticlockwise direction. This generates 2^j potential contour configurations around which to integrate in the new domain. In the case of I10, there are eight such contours and two of them avoid the threshold singularities. Therefore, the Taylor expansion of the complex-mapped subsector in the θ_j parameters can be integrated to yield approximations that evaluate to give precise and accurate numerical results at all kinematic points, even those over threshold. The successful application of this idea will be presented in Chapter 9.

8.8 Summary: The Necessity of the TayInt Algorithm

Of course, that was just one subsector of one Feynman integral at one order in ϵ. This successful idea must be converted into a program capable of producing such accurate and kinematically generalisable algebraic approximations, for any subsector of any Feynman integral at any order in ϵ, to meet the objectives of this thesis. This requires the capability to determine the contour configurations which produce subsector integrands well suited for a Taylor expansion in any kinematic region in an *automated* way. Hence, an *algorithm* is required. The production, development, refinement, programming and testing of this algorithm constitutes the particular challenge of calculating two-loop Feynman integrals up to the four-point and elliptic level by means of a Taylor expansion, all of which demanded considerable mathematical and computational research. Now that the presentation of the general difficulty in computing two-loop Feynman integrals has been concluded with the first half of this thesis, the particular difficulty of calculating them in the novel algorithmic way pioneered by this thesis will be explained in the upcoming chapters.

9 | The Conceptual TayInt Algorithm

9.1 Introduction

Now that the way to generate a representation for the subsectors of Feynman integrals which facilitates a well-converging Taylor expansion in the Feynman parameters has been found, it needs to be formalised into a method that can be applied to multiple integrals, regions of phase space and orders in ϵ. This leads to the first realisation of the objective of generating an algebraic integral library for Feynman integrals. As such, this is a lengthy undertaking and the backbone of the method to do so is as follows:

1. Input an integral.

2. Reduce the integral to a quasi-finite basis introduced in Refs. [65,66], such that the divergences are in the coefficient of the simplest integrals. An automated script using the libraries of the publicly available program Reduze [66,107,117] performs this, as described in Chapter 5.

3. For those basis integrals that are not known in analytic form, a decomposition into subsectors with smoother integrands is carried out. These are obtained using the program SecDec 3.0.9 [51–54], without its contour deformation option. The subsector integrands are analytic within the integration region but may contain integrable singularities over thresholds and at upper integration boundaries. The method of sector decomposition was described in detail in Chapter 6.

4. Use a conformal mapping to move the singularities outside of the region of integration as far away as is possible. This is done in Mathematica [118]. The structure of conformal mappings is such that the proximity between the integration regions and the singular behaviour is minimised.

5. a) To produce a result valid below the kinematic thresholds, the integrand is Taylor expanded around the midpoint of the integration region and integrated over the Feynman parameters. This is all done in FORM [119, 120].

   b) To produce a result valid over thresholds, there is a separate algorithm which determines how to calculate integrals in each kinematic region that is over a

threshold. This algorithm is implemented in Mathematica. The subsectors are first mapped onto the complex half plane. The algorithm then determines which configuration to use for each sector, that is, which contour orientation to use for the multiple variable integration and how to partition the subsequent region into smaller pieces. The Taylor expansion and integration are then performed on the new integrands specified by TayInt. The expansion points are always the midpoints of each interval.

This is the first proof-of-concept of the TayInt algorithm, henceforth referred to as the conceptual TayInt algorithm. It will be developed and finalised in Chapter 11 and simplified and automated into a Python program in Chapter 12. In this Chapter, the first implementation of the concept is described and demonstrated- as follows. In Section 9.2, each step of the conceptual TayInt algorithm is described in more detail. Section 9.3 gives an analysis of the virtues of each step of the conceptual TayInt algorithm. Numerical evaluations of the algebraic approximations for two-loop integrals relevant for phenomenological applications will be given in Chapter 13, after the final TayInt algorithm has been presented in Chapter 11.

9.2 The Algorithm

The form of a generic Feynman loop integral G in an arbitrary number of dimensions D was given and discussed in Eqs. (4.12) (4.14) of Chapter 4. After Feynman parametrisation and integration of the loop momenta this takes the general form stated in Eq. (4.41) of the same chapter. In order to implement the idea developed in Chapter 8 and calculate generic Feynman parametrised loop integrals as rational functions of the kinematic parameters, several steps are required. These are described in what follows.

9.2.1 U1: The Quasi-Finite Basis

The first universal step (U1) in the first implementation of the TayInt algorithm is to express the given Feynman integral G as a superposition of finite Feynman integrals G^{F} multiplying factorised poles in ϵ. These finite integrals are either defined in a shifted number of dimensions about $D = 4 - 2\,\epsilon$, have propagator powers greater than unity, or both. The combination of these quasi-finite integrals which yields the original integral is found via integration-by-parts [110,111], Lorentz invariance identities [7] and the Laporta algorithm [105]. In practice, an automated script steers the performance of all necessary steps in the program Reduze [66,107] towards the generation of the quasi-finite basis, as described in Chapter 5. The user must input the integral to be reduced and the integrals that are preferred for the finite basis. In the output, the divergences are restricted to the coefficients of the simpler integrals in the basis, so that the most complicated integral is always finite. Finding an optimal basis partially requires making an educated guess. The guiding principles are to express ultraviolet divergences in terms of vacuum integrals and to relate subdivergences to sub-graphs of the original integral under consideration.

9.2.2 U2: The Sector Decomposition

Once the original Feynman integral has a quasi-finite basis representation, the analytically unknown integrals in this basis are written in terms of their Feynman parametrisation and then decomposed into subsectors which have smoother integrands. These subsector integrals are the building blocks of the rest of the TayInt calculation. Their improved smoothness is achieved using sector decomposition [42–45]. Thus, the second universal step (U2) in the conceptual TayInt algorithm is to perform the sector decomposition of the integrals G^{F} in the quasi-finite basis by passing them to version 3 of the program SecDec [53]. Therein, the strategy `G2`, based on Ref. [121, 122] and combined with the Cheng-Wu theorem [123, 124] in Ref. [53], is used to yield sectors of the form

$$G_l^{\mathrm{F}}(\{q\},\{m\}) = \prod_{\tilde{j}=2}^{N} \int_0^1 \mathrm{d}t_{\tilde{j}}\, t_{\tilde{j}}^{A_{l\tilde{j}}-B_{l\tilde{j}}\epsilon} \frac{\mathcal{U}_l^{N_\nu-(L+1)D/2}\left(\vec{t}_{\tilde{j}}\right)}{\mathcal{F}_l^{N_\nu-LD/2}\left(\vec{t}_{\tilde{j}},\{q\},\{m\}\right)}, \quad l=1,\ldots,r\ , \tag{9.1}$$

where N is the number of propagators, $N_\nu = \sum_{\tilde{j}}^{N} \nu_{\tilde{j}}$, L the number of loops, $\nu_{\tilde{j}}$ the propagator powers and r is the number of subsector integrals. $A_{l\tilde{j}}$ and $B_{l\tilde{j}}$ are numbers independent of the dimensional regulator ϵ. A detailed explanation of this equation was given in Chapter 6. There are only $N-1$ integrations in total because the first Feynman parameter t_1 is always integrated out with the δ-distribution. By construction, the deterministic algorithm results in integrands of the type

$$\mathcal{U}_l = 1 + u\left(\vec{t}_{\tilde{j}}\right) \tag{9.2}$$

$$\mathcal{F}_l = s_1 + \sum_\beta s_\beta f_\beta\left(\vec{t}_{\tilde{j}}\right)\ , \tag{9.3}$$

where $u\left(\vec{t}_{\tilde{j}}\right)$ and $f_\beta\left(\vec{t}_{\tilde{j}}\right)$ are polynomials in the Feynman parameters $t_{\tilde{j}}$ and $s_1, s_\beta \in \{\{q\},\{m\}\}$ are kinematic invariants including masses. If the integral were not finite, the singular behaviour would now be contained entirely in the exponents $A_{l\tilde{j}}$ of Eq. (9.1). As the first Feynman parameter has been integrated out, the mapping $t_{\tilde{j}} \to t_j$ is performed, where j runs from 0 to $J-1$. This ensures a sensible hierarchy of parameters. The full integral can then be written in terms of its subsectors

$$G^{\mathrm{F}}(\{q\},\{m\}) = \frac{(-1)^{N_\nu}}{\prod_{j=1}^{N}\Gamma(\nu_j)}\Gamma\left(N_\nu - LD/2\right)\sum_{l=1}^{r} G_l^{\mathrm{F}}(\{q\},\{m\})\ . \tag{9.4}$$

Knowing that the integrals to be computed are finite, a sector decomposition might seem unnecessary. However, it is highly desirable for three reasons. Firstly, it ensures that there are no discontinuities due to the Feynman parameters *within the region of integration*, meaning that below any thresholds the subsectors already have a well-converging Taylor expansion. Secondly, the untangling of the Feynman parameters means that when thresholds are crossed, the resulting singularities can be more clearly seen and avoided by an appropriate contour configuration and partitioning. Thirdly, in the final TayInt algorithm the dependence on Reduze will be removed. This is described in

Chapter 11 and demonstrated in Chapter 13 by presenting TayInt approximations for divergent integrals. The sector decomposition is then crucial for the disentangling of the singularities in the Feynman parameters.

9.2.3 BT1-2: The Conformal Mapping and Taylor Expansion and Integration

For below-threshold calculations, there still exist singular behaviours outside of the integration region. Thus, the first below-threshold step (BT1) in the conceptual TayInt algorithm is to maximise the distance to the nearest point of non-analyticity and so maximise the accuracy of the expansion. This is achieved by exporting the finite subsector integrands to Mathematica [118] and applying conformal mappings,

$$t_j = \frac{a y_j + b}{c y_j + d}\,, \tag{9.5}$$

where y_j are the Feynman parameters in the conformal space. In all the cases considered in the course of this thesis, for an integrand decomposed into r subsectors and containing J Feynman parameters the optimal mapping has taken the following form in the case of the first subsector,

$$t_j = \begin{cases} \frac{-y_j-1}{y_j}, j \in \{0, ...J-2\} \text{ for } l = 1 \\ y_j, j = J-1, \text{ for } l = 1 \\ \frac{-y_j-1}{y_j}\,, j \in \{0, ...J-1\} \text{ for } l \in \{2, ..., r\}\ . \end{cases} \tag{9.6}$$

For the examples considered, the final Feynman parameter in the first sector was never mapped, as this parameter always appeared in the form $(1 + t_{J-1})$ in the denominators of the sectors. Thus, it is of no benefit to stretch the surface in that direction. The second below-threshold step (BT2) in the conceptual TayInt algorithm is to perform the Taylor expansion and integration of the integrand, the relative simplicity of which is best suited to using FORM [119, 120]. To this end, a FORM procedure for Taylor expanding and integrating functions of the form of the subsectors in general was written. The integration of the subsectors in the y_j is carried out from $y_j(0)$ to $y_j(1)$. This yields results for Feynman integrals as rational functions of the kinematic scales valid everywhere below threshold. The precision is controlled by the order of the expansion. The conceptual TayInt algorithm has now been presented up to step BT2, shown in bold in Table 9.1.

9.2.4 Going Over Threshold

In the kinematic region above the lowest mass threshold of a particular integral, the integrands contain discontinuities on the real axis which prevent a Taylor expansion from converging. Thus, the conceptual TayInt algorithm returns to the result of U2, specifically the subsector integrands $\tilde{G}_l^{\mathrm{F}}$, the multivariate integrands of G_l^{F}. The Feynman $+i\delta$ prescription of Eq. (4.14) is then implemented in Mathematica. This is done by

Table 9.1: Summary of the individual steps of the conceptual TayInt algorithm, with the labels of the steps U1-BT2 presented so far typeset in boldface.

U1: reduce the Feynman Integral to a quasi-finite basis	
U2: perform a sector decomposition on the finite integrals in the basis	
Below threshold	Over threshold
BT1: $t_j \to y_j$	OT1: $t_j \to \theta_j$, generate $\mathcal{K}$
BT2: Taylor expand the integrand and integrate	OT2: find optimum $\Theta_{o(0),\ldots,o(J-1)}$
	OT3: perform one-fold integrations
	OT4: post-integration, find optimum $\Theta_{o(0),\ldots,o(J-2)}$
	OT5: determine partition $\mathcal{P}_j$
	OT6: Taylor expand and integrate

mapping the multivariate integrands of G_l^{F}, onto complex half planes. The conceptual TayInt algorithm then determines the contour configuration which avoids the poles in each kinematic region that is over a threshold. The outline of the over-threshold part of the algorithm, which is implemented in each kinematic region that is above a threshold, is as follows:

1. Implement the Feynman $+\mathrm{i}\delta$ prescription of Eq. (4.14) by transforming the sub-sector integrands $\tilde{G}_l^{\mathrm{F}}$ as

$$\tilde{G}_l^{\mathrm{F}}(t_j) \to \tilde{G}_l^{\mathrm{F}}(t_j') = \tilde{G}_l^{\mathrm{F}}\left(\frac{1}{2} + \frac{1}{2}\, e^{\mathrm{i}\theta_j}\right) , \tag{9.7}$$

$$t_j' = \frac{1}{2}\left(1 + \cos\theta_j + i \sin\theta_j\right) , \tag{9.8}$$

   with $j \in \{0, \ldots, J-1\}$. The mapping is chosen such that the real part stays between 0 and 1 and the imaginary part parametrises a contour around a Landau singularity.

2. Find the optimum contour configuration for each θ_j with end points 0 and $\pm\pi$, the combination of which is denoted by $+$ or $-$, respectively. On this contour configuration, find the optimum variable θ_j^* to integrate exactly and hence the optimum post-integration contour configuration, if exact integration is possible.

3. Determine the optimum n-fold partitioning $\mathcal{P}_j = \{(l,h)_1, ..., (l,h)_n\}_j$ of the integrals in θ_j,

$$\int_0^{\pm\pi} \mathrm{d}\theta_j = \sum_{k=1}^{n} \int_{l_{k,j}}^{h_{k,j}} d\theta_j , \tag{9.9}$$

   with $h_{n,j} = \pm\pi$ and $l_{1,j} = 0$, to use for the Taylor expansion of each sector integrand.

4. Perform the Taylor series expansion in each partition, about the points $e_{j,k} = \frac{1}{2}(l_{k,j} + h_{k,j})$ up to order p in each sector

$$G_l^{\mathrm{F}}(\{q\},\{m\}) \approx T_l^{\mathrm{F}}(\{q\},\{m\}) = T_l^{\mathrm{F}(0)}(\{q\},\{m\}) + \cdots + T_l^{\mathrm{F}(p)}(\{q\},\{m\}) \ , \tag{9.10}$$

and estimate the uncertainty of the full result by comparing the relative size of the contribution of the p-th order to the full Taylor series expansion. Adding the contribution from each p-th order expanded sector in quadrature this reads,

$$\max[G^{\mathrm{F}}(\{q\},\{m\}) - T^{\mathrm{F}}(\{q\},\{m\})] < \frac{\sum_{l=1}^{r}(T_l^{F(p)}(\{q\},\{m\}))^2}{\sum_{l=1}^{r} T_l^{F}(\{q\},\{m\})} \ . \tag{9.11}$$

Because of the use of exact one-fold integrations where possible, and the partitioning of the surface, the over-threshold algorithm combines algebraic and analytic manipulations. This requires flexibility. Therefore, it is implemented in Mathematica. This is the first conceptual implementation of the crucial over-threshold part of the TayInt algorithm, the most important part of the TayInt algorithm. It it will be significantly developed in Chapter 11 to produce the final algorithm. Nevertheless, the core ideas of choosing a contour configuration and partitioning the integrand remain throughout.

9.2.5 OT1: Set-up

To elaborate upon this core over-threshold part of the algorithm, its first step (OT1) is to transform the Feynman parameters of the r subsectors. This is done according to the rule, $t_j \to \frac{1}{2} + \frac{1}{2}\exp(\mathrm{i}\theta_j)$. A representative sample of the kinematic region in which the results are to be valid is also generated. This is a nested list of values for the β scales in the integral at γ points in the kinematic region within which the user desires results, $\mathcal{K} = \{\{s_1,\ldots,s_\beta\}_1,\ldots,\{s_1,\ldots,s_\beta\}_\gamma\} = \{\mathcal{K}_1,\ldots,\mathcal{K}_\gamma\}$.

9.2.6 OT2-4: Choosing the Contour Configurations

After that, the second over-threshold step (OT2) uses the mean absolute value of the θ_j derivatives (MAD: $\bar{m}_l$), with the kinematic invariants set to the sample values,

$$\bar{m}_l(\Theta^A_{o(0),\ldots,o(J-1)}) = \frac{1}{A}\sum_{a=1}^{A}\frac{1}{\gamma}\sum_{i=1}^{\gamma}\mathrm{Abs}\left[\frac{1}{J}\sum_{j=0}^{J-1}\left(\frac{\partial}{\partial\theta_j}\tilde{G}_l^{\mathrm{F}}\left(\theta_j,\mathcal{K}_i\right)\right)\right]\Bigg|_{\{\theta_j\}\to\Theta^a_{o(0),\ldots,o(J-1)}} \ . \tag{9.12}$$

Note that the mean is also taken over the kinematic sample and the points for the θ_j inserted along the surface, $\Theta^A_{o(0),\ldots,o(J-1)} \subset \Theta_{o(0),\ldots,o(J-1)}$. The MAD is calculated for all possible J-variable complex surfaces in the θ_j. These surfaces are the $\Theta_{o(0),\ldots,o(J-1)}$, where $o(j) = \pm$ is the orientation of the jth contour. Each surface is classified by replacing the θ_j variables by A points along it, $\Theta^A_{o(0),\ldots,o(J-1)}$, in the mean absolute

derivative with respect to the θ_j. Scanning the surfaces $\Theta_{o(0),\ldots,o(J-1)}$ using the MAD is done to determine which contour orientation, for example Θ_{+-+} for $J = 3$ yields the $\Theta_{o(0),\ldots,o(J-1)}$ best suited for an expansion. The plus and minus signs indicate the direction of motion around the contour. The separation of the points in the $\Theta^A_{o(0),\ldots,o(J-1)}$ in the MAD is set to a default value of 0.1 which is sufficient to determine the optimum $\Theta_{o(0),\ldots,o(J-1)}$ surface. However, this value can be varied by the user.

The scanning is done in two stages. Firstly, the MAD in the corners of the $\Theta_{o(0),\ldots,o(J-1)}$ surfaces is calculated, as it is here that the change is always the most substantial. Any contour configurations which yield extreme changes at the corners are discarded. Then, the MAD for the remaining $\Theta_{o(0),\ldots,o(J-1)}$ surfaces is calculated, with the corners excluded. This is because the larger changes in the corners mask the changes in the bulk of the $\Theta_{o(0),\ldots,o(J-1)}$ surfaces. The $\Theta_{o(0),\ldots,o(J-1)}$ which minimises the MAD in the second stage is then selected.

As this is a J-fold surface, the third over-threshold step (OT3) is to perform all possible one-fold integrations in the θ_j exactly, without using an integrand expansion. A time limit is imposed on this operation. If the limit is exceeded, the conceptual TayInt algorithm automatically reverts to the $\Theta_{o(0),\ldots,o(J-1)}$ from OT2.

Next, the fourth over-threshold step (OT4) is to determine the optimum post-integration surface, $\Theta_{o(0),\ldots,o(J-2)}$. To achieve this, all the one-fold exact integrations are performed and all resultant $J-1$ variable integrands examined, using the two-step surface scanning process with the MAD. The $\Theta_{o(0),\ldots,o(J-2)}$ which minimises the MAD is selected. If the mean absolute derivatives of each of the possible J or $J-1$ surfaces are equally smooth (within a relative difference of 0.01) then the process stops. This is because all the possible surfaces are then too similar and the potential for further optimisation is negligible. Provided there is another J variable contour which has a MAD within a relative difference of 0.01, the 2^{nd} best surface is taken and the procedure starts again. The MADs of each surface, with and without exact integration, are finally compared and the surface with the overall minimum MAD is selected. In the vast majority of cases the post-integration surfaces are chosen, if an exact integration is possible.

9.2.7 OT5: Partitioning

The fifth over-threshold step (OT5) determines the optimal way to partition the surface into sections $\mathcal{P}_j$ within which the expansions are carried out. The conceptual TayInt algorithm determines the number of partitions to use and the size of each partition. This is done by using the MAD to calculate the relative size of the fluctuations in each section of the surface. A suitable partitioning is then chosen, i.e., the more fluctuations, the greater the number of partitions; and the denser the fluctuations, the smaller the section enclosing that region of the surface. If no exact integration can be performed, TayInt can use two more orders or twice as many partitions to maintain the same degree of accuracy.

9.2.8 OT6: Taylor Expansion and Integration

In summary, the core over-threshold part of the conceptual TayInt algorithm makes a complex mapping in several variables. It then determines the optimum pre- and post-exact integration contour configuration. Finally, it determines the optimum partitions for the integrand expansion. This is done for each kinematic region that is over a threshold. The resulting integrands of each sector, $\tilde{G}_l^F(\theta_0, \ldots, \theta_{J-1})$ are then expanded and integrated in the sixth over-threshold step (OT6),

$$T_l^F(\{q\},\{m\}) = \prod_{j=0}^{J-1} \left(\sum_{k_j=1}^{n_j} \int_{l_{k_j}}^{h_{k_j}} \mathrm{d}\theta_j \sum_{s_j=0}^{m_j} \frac{(\theta_j - e_{j,k})^{s_j}}{s_j!} \frac{\partial^{s_1+\ldots+s_J}}{\partial\theta_1^{s_1} \ldots \partial\theta_{J-1}^{s_{J-1}}} \tilde{G}_l^F(\{q\},\{m\},e_1,\ldots,e_{J-1}) \right), \tag{9.13}$$

where $e_{j,k}$ are the expansion points, the midpoints of each partition, m_j the order of the expansion and n_j the number of partitions, in each parameter θ_j. The conceptual TayInt algorithm calculates the Taylor expansion of the integrand and then integrates this expansion to produce an approximation of the integral as a function of the expansion points and integration boundaries. Then all the relevant expansion points and boundaries are inserted to generate the approximation in each partition. Next, these approximations are summed to give the result for each subsector. Adding up the approximations for each sector in each kinematic region generates a systematic approximation for the full finite Feynman integral in terms of rational functions of the kinematic scales that is valid everywhere in that region. Thus, the desired library of systematic approximations is obtained. It can be used to produce numerical approximations in all kinematic regions, above and below mass thresholds. The precision of the approximation is controlled by the order of the expansion and the resolution of the partitioning.

Finally, to estimate the uncertainty in the TayInt calculation, the truncation error in the Taylor expansion is calculated. To do this, the highest order contribution of all T_l^F, where $s_j = m_j$, are considered, summed in quadrature and divided by the full result, T^F, where s_j runs from 0 to m_j. Due to the fact that an integration over all θ_j parameters is performed, including the parameter which gives the maximum contribution to the uncertainty, all possible sources of uncertainty are taken into account. The resulting uncertainties for each sector are then added in quadrature to produce the final uncertainty estimate in the result. Note that the uncertainty will be overestimated in the vicinity of any kinematic point at which one of the sectors evaluates to a numerical zero by TayInt, meaning that the $\tilde{G}_l^F(\theta_0, \ldots, \theta_{J-1})$ is oscillatory. Nevertheless it always constitutes an overestimation of the uncertainty when less reliable. Moreover, the uncertainty estimate is always highly conservative. This is because the pth order of the Taylor expansion is used to estimate the truncation errors in the results calculated using an expansion up to order p, rather than the order $p+1$. This method of error estimation remains unchanged in the final TayInt algorithm. At the end of this subsection, all of the steps in the conceptual TayInt algorithm have been filled in, as is summarised in Tab. 9.2.

Table 9.2: Summary of the individual steps of the conceptual TayInt algorithm, which have all now been presented.

U1: reduce the Feynman Integral to a quasi-finite basis	
U2: perform a sector decomposition on the finite integrals in the basis	
Below threshold	Over threshold
BT1: $t_j \to y_j$	**OT1**: $t_j \to \theta_j$, generate $\mathcal{K}$
BT2: Taylor expand the integrand and integrate	**OT2**: find optimum $\Theta_{o(0),\ldots,o(J-1)}$
	OT3: perform one-fold integrations
	OT4: post-integration, find optimum $\Theta_{o(0),\ldots,o(J-2)}$
	OT5: determine partition $\mathcal{P}_j$
	OT6: Taylor expand and integrate

9.3 Discourse on the Method

To facilitate a deeper understanding of the aforementioned method, the different steps are illustrated for the integrals in Fig. 9.1, their characteristics being listed in Tab. 9.3. Results for all these integrals will be provided in Chapter 13. The finite sunrise $S14^{01220}$ [125] and triangle graph T41 [125] serve as examples to illustrate and explain the conceptual TayInt algorithm. The integrals I10 [92], I21 [92], I246 [126] and I39 appear in the two-loop amplitudes [32,60,127] for Higgs-plus-jet production in gluon fusion, mediated through a massive top quark loop. They demonstrate that the conceptual TayInt algorithm is applicable to complicated multi-loop, multi-scale integrals, indicating the rich potential of the idea. The number of Feynman parameters quoted in Tab. 9.3 is counted after performing the integration of the δ-distribution of Eq. (4.12). In what follows and where given, the powers of each propagator are denoted by superscripts, i.e. X^{ijk}. The kinematic invariants s and u are defined as $s = (p_1+p_2)^2$ and $u = (p_2-p_3)^2$, respectively.

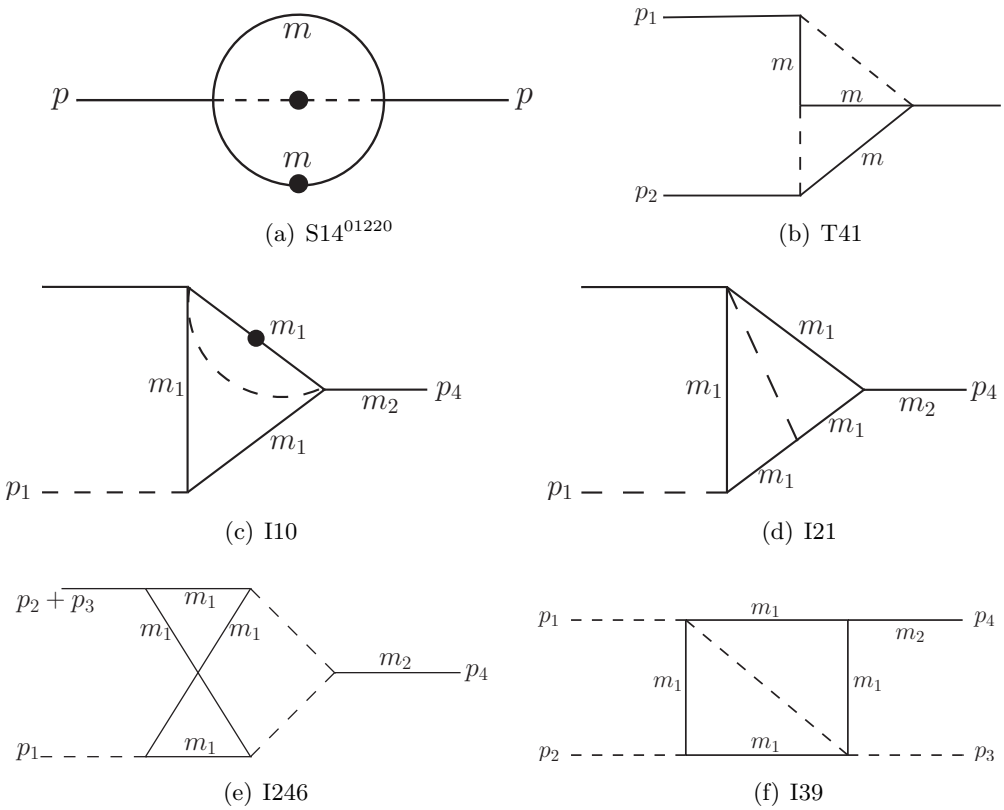


Figure 9.1: The two-loop finite sunrise S14^{01220} and triangle T41, I10, I21 graphs for which analytical results are available [92, 125]. The non-planar I246, (a) and The box-type integral I39, (c), are so far unknown analytically. Dashed lines indicate massless, solid internal lines massive, and dots squared propagators. Solid external lines denote, where indicated, massive and else off-shell particles.

Table 9.3: Properties of the integrals corresponding to the diagrams depicted in Fig. 9.1. The stated numbers of Feynman parameters correspond to the dimensionality of the expansion and integration steps of TayInt.

Graph	Scales	Feynman parameters	Subsectors
S14^{01220}	2	2	3
T14	2	4	28
I10	3	3	8
I21	3	4	16
I39	4	4	16
I246	3	6	36

9.3.1 Universal Steps

U1: The Quasi-Finite Basis

1. As an example of the step U1 of the TayInt algorithm, the divergent sunrise integral $\mathrm{S14}^{01110}$ [125] is written in terms of the finite integrals $\mathrm{S14}^{01220}$, $\mathrm{S14}^{01320}$ and the tadpole $\mathrm{S6}^{30300}$,

$$\begin{aligned}\mathrm{S14}^{01110} = &\frac{8\,m^2\,(p^2-4\,m^2)\,(p^2+2\,m^2)}{(-3+D)(-8+3\,D)(-10+3\,D)}\cdot\mathrm{S14}^{01320}\\ &+\frac{((4-D)\,p^4+(-5+D)\,8\,m^4+(18-5\,D)\,4\,p^2m^2)}{(-3+D)(-8+3\,D)(-10+3\,D)}\cdot\mathrm{S14}^{01220}\\ &-\frac{16\,m^4\,((-4+D)\,p^2+2\,(-24+7\,D)\,m^2)}{(-3+D)(-4+D)^2(-8+3\,D)(-10+3\,D)}\cdot\mathrm{S6}^{30300}\ ,\end{aligned} \tag{9.14}$$

where the poles in ϵ can be seen as the $(-4+D)^{-1}$ terms. As advertised, the divergences are confined to the coefficient of the simplest integral. This was discussed fully in Chapter 5.

U2: The Sector Decomposition

2. As an example of step U2 of the conceptual TayInt algorithm, the $\mathcal{O}(\epsilon^0)$ coefficient of the first subsector of $\mathrm{S14}^{01220}$ reads

$$\mathrm{S14}^{01220}_1 = -\prod_{j=0}^{1}\int_0^1 dt_j \frac{2}{m^2(1+t_0+t_0t_1)\,((1+t_1)(1+t_0+t_0t_1)+t_0t_1p^2/m^2)}\,, \tag{9.15}$$

and has two Feynman parameters, t_0, t_1 and two scales, p^2 and m^2. The $\mathcal{O}(\epsilon^0)$ coefficient of the first and second subsector of the I10 [92] integral read

$$\begin{aligned}\mathrm{I10}_1 = &\prod_{j=0}^{2}\int_0^1 dt_j((1+t_0+t_1+t_0t_2+t_1t_2)\,(-m_2^2t_0-ut_1+m_1^2(1+t_0^2(1+t_2)\\ &+t_1^2(1+t_2)+t_1(2+t_2)+t_0(2+t_2+t_1(2+2t_2)))))^{-1}\,,\end{aligned} \tag{9.16}$$

$$\begin{aligned}\mathrm{I10}_2 = &\prod_{j=0}^{2}\int_0^1 dt_j((1+t_0+t_1+t_2+t_1t_2)\,(t_0(-u-m_2^2t_1)+m_1^2(1+t_0^2+t_2+t_1^2\\ &(1+t_2)+t_1(2+2t_2)+t_0(2+t_2+t_1(2+t_2)))))^{-1}\ .\end{aligned} \tag{9.17}$$

They have three Feynman parameters and three kinematic scales, m_1, m_2 and u.

9.3.2 Below-Threshold Steps

The most precise way of computing the resulting subsector integrals was investigated by comparison. Explicitly, the approximations obtained using various ways of expanding integrands were checked against the literature results for analytically known Feynman integrals. In particular, a comparison of expansions into Taylor series, geometric series, reverse Padé approximations, Chebyshev and Gegenbauer polynomials exposed the Taylor expansion as having the best blend of accuracy and speed given a variety of test cases. In Fig. 9.2, the relative difference between the actual $\mathrm{S14}_1^{01220}$ subsector integrand and its fifth-order Taylor expansion is plotted. While the expansion is rather accurate around the expansion point $(t_1, t_2) = (\frac{1}{2}, \frac{1}{2})$, the differences between an ordinary Taylor expansion and the actual integrand can become large close to the edges of the integration region. In the case of $\mathrm{S14}_1^{01220}$, these amount to roughly 1%.

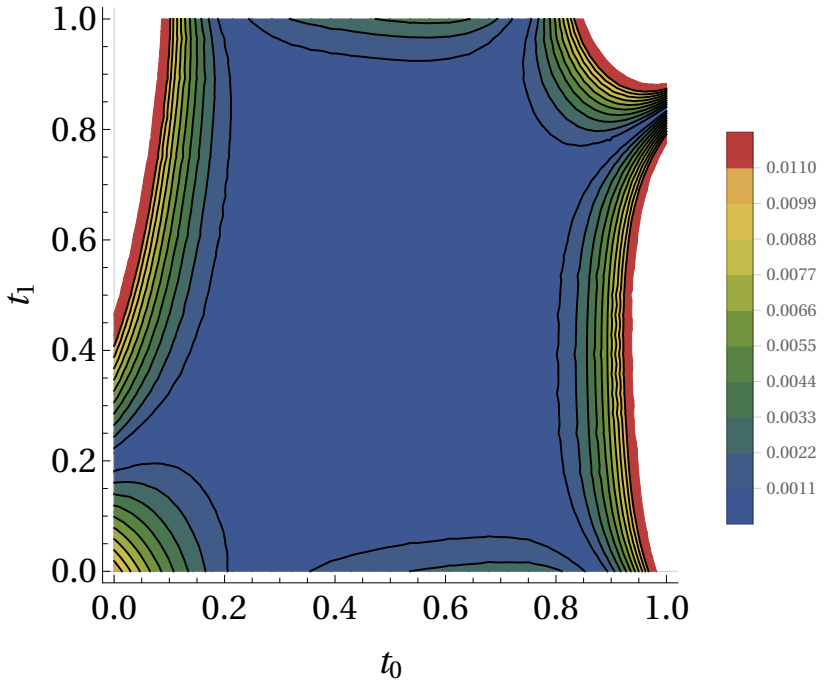
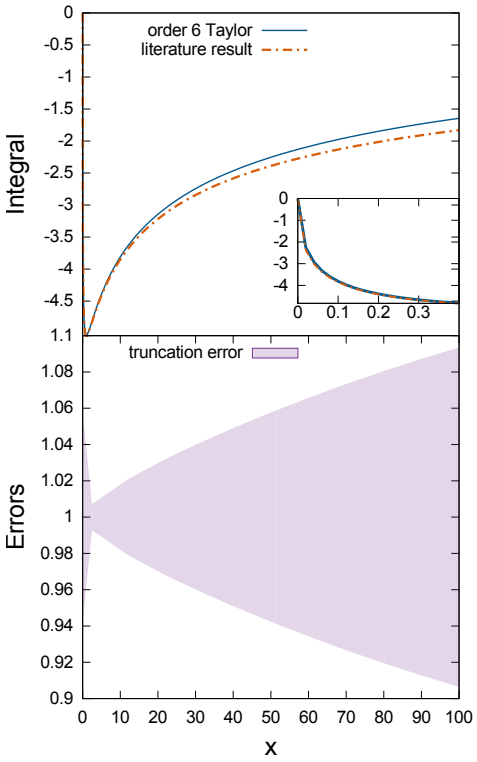
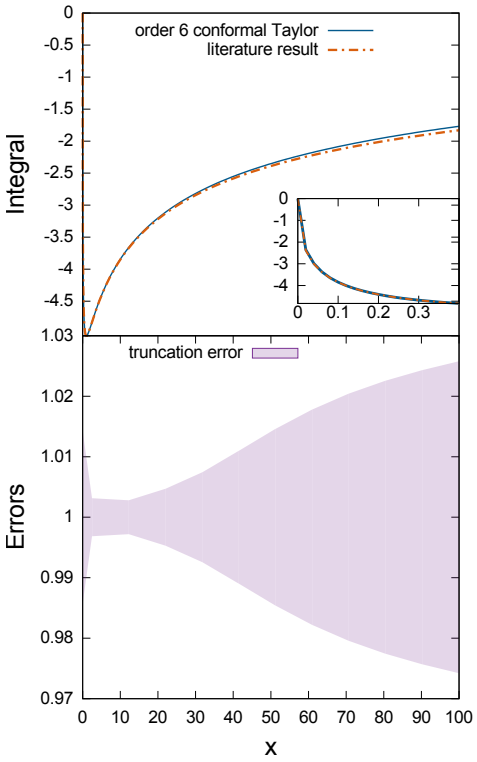


Figure 9.2: A contour plot of the relative difference between an ordinary sixth-order Taylor expansion and the actual integrand of $\mathrm{S14}_1^{01220}$. The Taylor expansion is around the point $(t_1, t_2) = (\frac{1}{2}, \frac{1}{2})$ and $p^2 = -\frac{1}{2}m^2$, $m^2 = 20000$ GeV^2.



(a)



(b)

Figure 9.3: A comparison between the integrated approximated result for the $\mathcal{O}(\epsilon^0)$ coefficient of T41 and the analytic result, using 9.3(a) an ordinary Taylor expansion on the integrand and 9.3(b) a Taylor expansion enhanced by a conformal mapping. The inlay figures have the same axis labels as the larger plots.

The Taylor expansion turns the rational function $R(t_j)$ into a polynomial $P(t_j)$ which can be integrated analytically. Thus, it is necessary to check that accurate results for integrals can still be obtained after expanding the integrand. To this end, the example integral T41 depicted in Fig. 9.1(b), the kinematic dependence of which is parameterised entirely by the dimensionless ratio x,

$$x = \frac{\sqrt{s + 4\,m^2} - \sqrt{s}}{\sqrt{s + 4\,m^2} + \sqrt{s}}\,, \tag{9.18}$$

was calculated by Taylor expanding the integrand to sixth order. In Fig. 9.3(a), the result is compared to the exact result of Ref. [125], with the truncation error of the Taylor expansion shown in the lower half of the plot. The Taylor-obtained approximation is plotted as a solid blue line in contrast to the literature result in a red dot dash line. The sixth-order truncation error is plotted below using a lilac band. It can be observed that the combination of expansion and integration is effective for calculating Feynman integrals via their subsectors for most values of x.

BT1-2: The Conformal Mapping, Taylor Expansion and Integration

For $x \approx 100$, the discrepancy between approximated and exact result roughly reaches an unacceptable 9%.

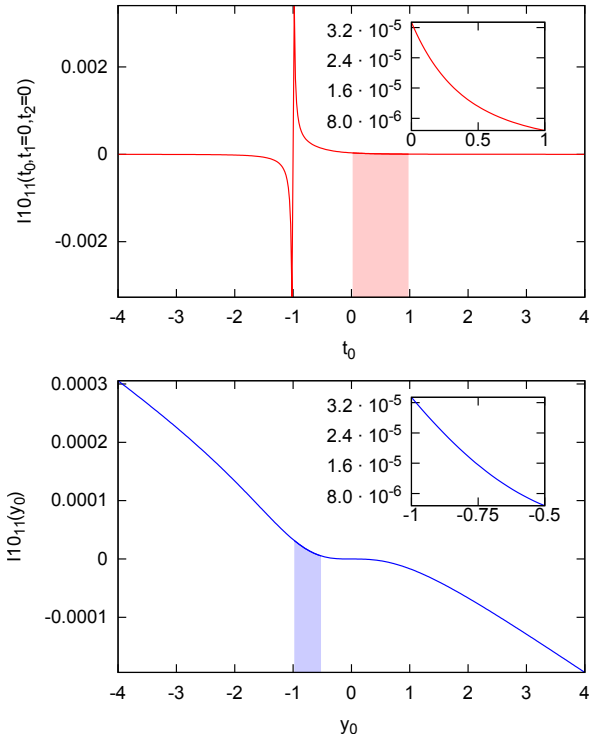


Figure 9.4: Plot of the one-dimensional integrand of Eq. (9.19), before and after the conformal mapping. The integration region is shaded and also shown in the inset plots. The masses are set to $m_1 = 173$ GeV and $m_2 = \frac{m_1}{\sqrt{2}}$.

To improve on the quality of the approximation, methods to maximise the distance to the nearest point of non-analyticity were investigated. The underlying rationale is that the convergence radius of a Taylor expansion is limited by the distance from the expansion point to the nearest point of non-analyticity. The latter are found in the region $t_j \in [0, -\infty]$ for the subsectors. The use of conformal mappings in the t_j, as detailed in Eqs. (9.5) and (9.6), maximises the convergence radius for the examples considered. As an example, the effect of a conformal mapping at the integrand level can be seen in Fig. 9.4. In the upper plot, the integrand of integral $\mathrm{I10}_1$ with two Feynman parameters set to 0,

$$\mathrm{I10}_1(t_0, t_1 = 0, t_2 = 0) = \prod_{j=0}^{2} \int_0^1 dt_j \frac{1}{(1+t_0)(-m_2^2 t_0 + m_1^2(1 + 2t_0 + t_0^2))} \,, \tag{9.19}$$

is shown. It has a point of non-analyticity outside the integration region, at $t_0 = -1$.

Applying the conformal mapping,

$$t_0 = \frac{-1 - y_0}{y_0} \,, \tag{9.20}$$

on the integrand, as detailed in Eq. (9.5), the region of integration changes to $y_0 \in [-1, -\frac{1}{2}]$ and the non-analytic point is stretched to infinity, as shown in the lower plot of Fig. 9.4. The plot insets zoom into the actual region of integration.

The quantitative improvement can be seen by comparing Figs. 9.3(a) and 9.3(b). In Fig. 9.3(a), the sixth-order Taylor expanded result for the $\mathcal{O}(\epsilon^0)$ coefficient of the T41 integral is compared to the exact result known from the literature [33]. In Fig. 9.3(b), a conformal mapping is applied to the T41 integrand before performing a Taylor expansion up to sixth order. For $x \approx 100$, the discrepancy between approximated and exact result decreases to less than 3% when using a conformal mapping.

To view the effect of applying a conformal mapping to a more complicated example, the ratio of the result computed with an integrand Taylor expansion up to sixth order and the result computed numerically using SecDec is plotted for the $\mathcal{O}(\epsilon^0)$ coefficient of the full integral I10, see Fig. 9.5. Error bars on the numerical results from SecDec are not plotted due to the high requested numerical accuracy of 10^{-8}. The ratio is plotted over a kinematic range below threshold, the result with a conformal mapping shown in green and the one without mapping in blue. The mean ratio between SecDec and the conceptual TayInt result over the plotted range is 1.00043 with the conformal mapping and 1.00134 without it. Using the conformal mapping therefore increases the precision by more than a factor of three.

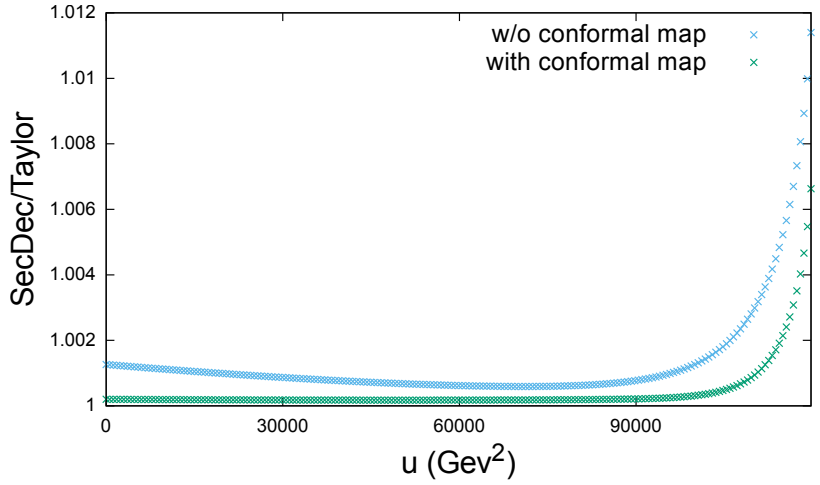


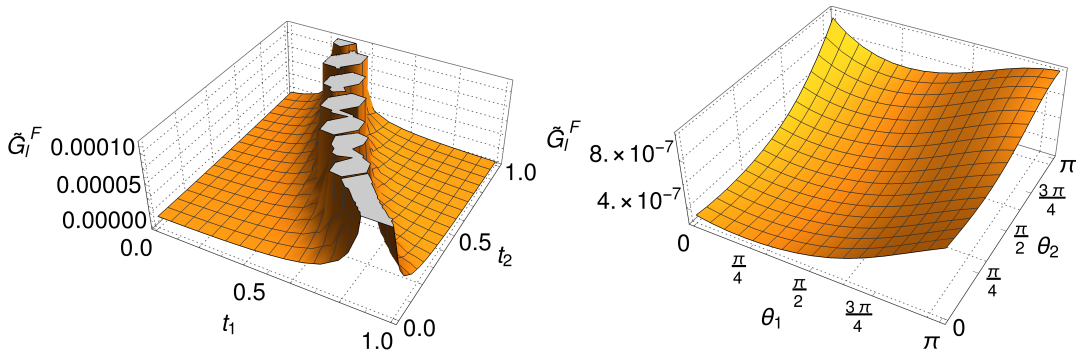
Figure 9.5: The ratio of the SecDec result and an ordinary Taylor expansion (6th order) are shown with (green) and without (blue) conformal mapping, for the $\mathcal{O}(\epsilon^0)$ coefficent of the integral I10. The scale u is below the $4m_1^2$ threshold and $m_2 = \frac{m_1}{\sqrt{2}}$, $m_1 = 173$ GeV.

9.3.3 Over-Threshold Steps

No further steps are required to calculate Feynman integrals below threshold. However, over thresholds there are integrable singularities in the subsectors, within the region of integration, which renders all steps beyond the sector decomposition moot.

OT1: Set-up

By transforming to the complex plane (OT1), these singularities can be avoided. In Fig. 9.6 a slice of the absolute value of $\mathrm{I10}_2$, Eq. 9.17, is plotted in t_0 without and in θ_0 with, a complex mapping. In Fig. 9.6(a), the integrand without an analytical continuation to the complex plane contains a series of threshold singularities. The ridges are cut for better comparison with Fig. 9.6(b). The absolute value of the complex integrand in Fig. 9.6(b) shows a smooth behaviour everywhere in the integration region. This demonstrates how integrable poles in the physical region can be avoided and confirms that an ordinary series expansion of the integrand in the Feynman parameters t_j is not sufficient to reproduce the actual result.



(a) (b)

Figure 9.6: A slice of the absolute value of the $\mathrm{I10}_2$ integrand at $\mathcal{O}(\epsilon^0)$ (a) without a complex mapping and (b) with a complex mapping and the contour orientation determined via TayInt, setting $\theta_0 = \frac{\pi}{2}$, $u = 5.44\, m_1^2$, $m_2 = \frac{m_1}{\sqrt{2}}$ and $m_1 = 173$ GeV. In (a) no reorientation of contours is possible as the surface is constrained to the real line.

OT2-4: Choosing the Contour Configurations

OT1 yields a version of the subsectors which can take different forms depending on the configuration of the complex contours. Thus, the optimum contour configuration from the possible $\Theta_{o(0),\ldots o(J-1)}$ surfaces must be determined. So must the optimum variable to integrate exactly, θ_j^*, if exact integration is possible. This yields the optimum post-integration surface, $\Theta_{o(0),\ldots,o(J-2)}$. Finding these optimum configurations is done in steps OT2-4. In Fig. 9.7, the absolute value of the integrand of $\mathrm{I10}_2$, Eq. 9.17, is

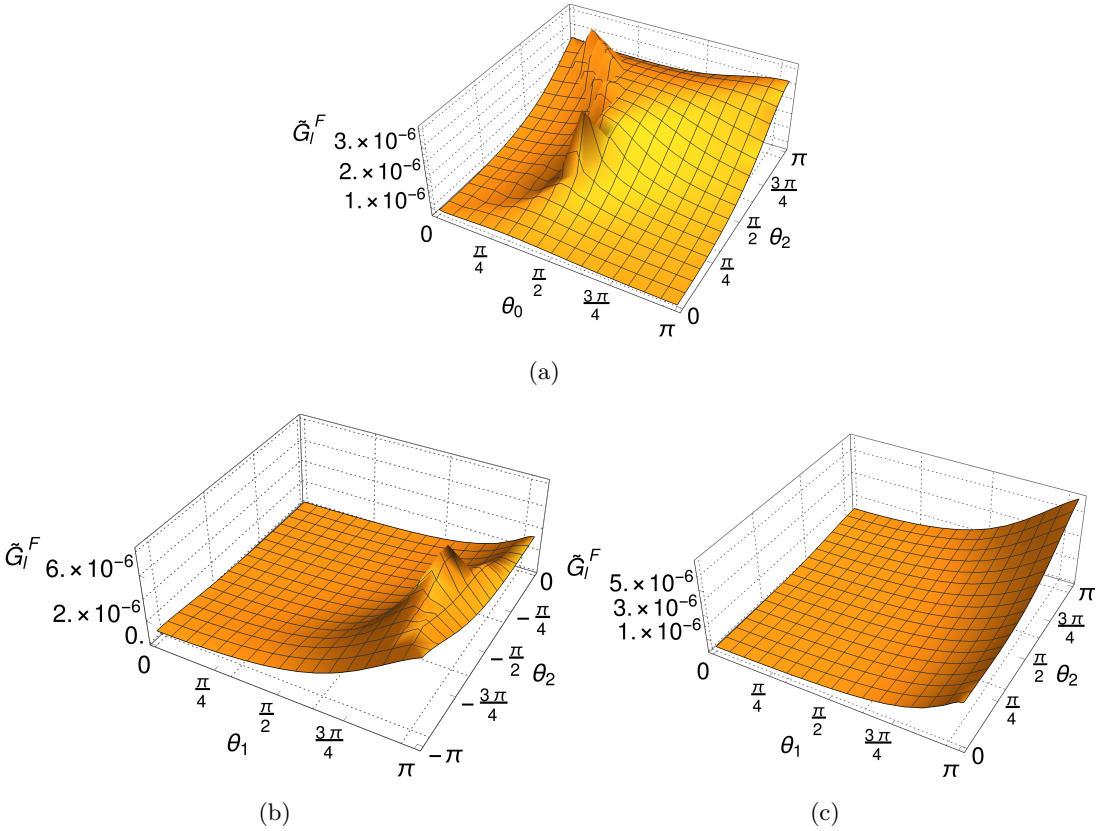


Figure 9.7: The absolute value of the $\mathrm{I10}_2$ integrand at $\mathcal{O}(\epsilon^0)$ is shown after a complex mapping and exact integration of one variable in three different configurations. In (a), the contour orientation is chosen by the algorithm but an arbitrary choice of integration variable is allowed. In (b), the exact integration variable is chosen by the algorithm but an arbitrary choice of contour is allowed. In (c), the contour and exact integration variable are chosen by the conceptual TayInt algorithm. The kinematic scales are set to $u = 5.44\, m_1^2$, $m_2 = \frac{m_1}{\sqrt{2}}$ and $m_1 = 173$ GeV.

plotted, with one integration performed exactly. There exist many possible pre- and post-integration contours. Not all of these contours are suitable for a Taylor expansion of the integrand. This is because, along the unsuitable contours, the integrand contains non-analytic structures within the region of integration. If such a contour was chosen, the algebraic result for that sector would not always converge at all kinematic points. The conceptual TayInt algorithm avoids these and selects a pre- and post-integration contour configuration which yields a smoothly behaved integrand. This has a well defined Taylor expansion. Figs. 9.7(a) and 9.7(b) illustrate this. The optimal result is achieved when both contours are determined by the conceptual TayInt algorithm, see Fig. 9.7(c).

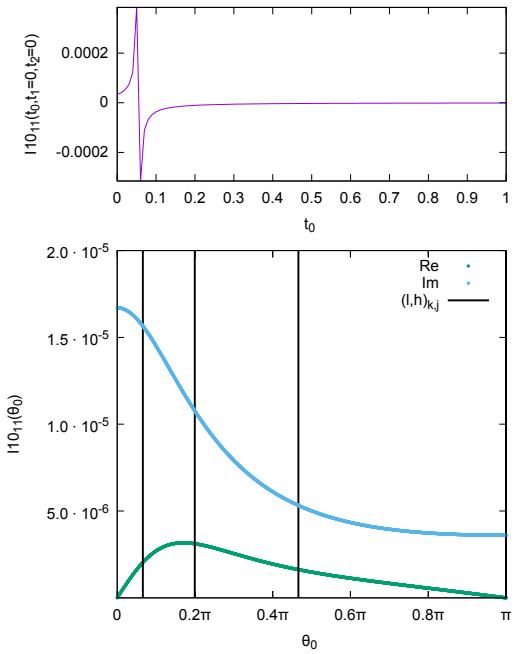


Figure 9.8: Plot of the one-dimensional integrand of Eq. (9.19), with chosen values $m_2 = 781.25$ GeV and $m_1 = 173$ GeV. The upper plot shows the integrand without the implementation of the Feynman $+i\delta$ prescription, the lower plot shows the integrand with it. The partitioning of the integral according to Eq. (9.9) is illustrated by the black lines.

After the analytic continuation of the subsector integrands is optimised, in OT5 large gradients along the edges of the complex surfaces(see e.g. the integrand surface along $\theta_0 = \pi$ of Fig. 9.7(c)) are addressed. To maximise the precision of the result, the surfaces are partitioned and expansions performed around the central value of each partition. The Taylor expanded sections are then integrated and combined to yield a result for the entire sector. The rationale behind the partitioning is demonstrated in Fig. 9.8, where the one-dimensional integrand of Eq. (9.19) is shown for kinematic invariants set to arbitrarily chosen over-threshold values $m_2 = 781.249$ GeV and $m_1 = 173$ GeV. In the upper plot, a discontinuity along the real line arising from the threshold can be observed. To remedy this problem, the integrand is analytically continued into the complex plane. This is described in Eq. (9.7) and demonstrated by showing the transformed real and imaginary part of the integrand in the lower plot of Fig. 9.8, in green and blue, respectively. Even though the integrand is now suited to a Taylor expansion, the gradient of the integrand is large for $\theta_0 \to 0$. Thus, the integration region is split according to Eq. (9.9), with the new integration boundaries $(l, h)_{k,j}$ marked by black lines in the bottom plot of Fig. 9.8. The expansion and integration is performed in each partition individually. The conceptual TayInt algorithm splits the integral such that within each partition the

gradient is small. Hence, the convergence of the Taylor expansion within each integral piece is faster than that of an expansion of the whole integrand. Using these partitions to control precision, the Taylor expanded and integrated approximation for the full Feynman integral is assembled in step OT6, the details of the calculation code are given in Appendix B.1.

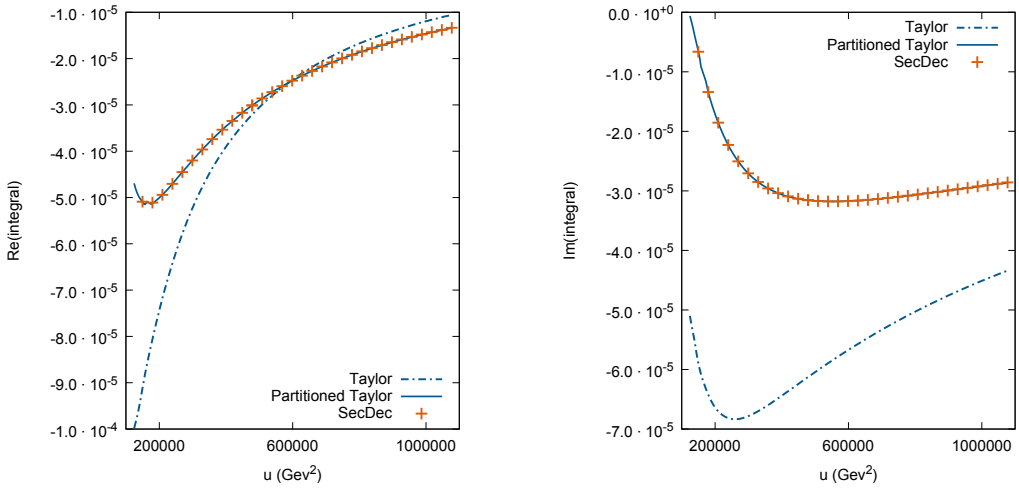


Figure 9.9: I10 integral at $\mathcal{O}(\epsilon^0)$ over the $4m_1^2$ threshold with and without partitions. The kinematic scales are $m_2 = \frac{m_1}{\sqrt{2}}$ and $m_1 = 173$ GeV.

For generating results valid over kinematic thresholds, one might ask why an ordinary Taylor expansion cannot be used after steps OT1-4. The importance of a partitioning is illustrated for the I10 diagram in Fig. 9.9. The result generated with a Taylor expansion without partitions is plotted as a dot-dashed blue line, the result obtained with partitions is shown as a solid blue line and the SecDec points are orange crosses. Without using partitions, the Taylor result converges slowly and a huge number of orders in the expansion would be required. But, if a particular integrand is extremely complicated, then there will be a limit on the order to which the Taylor expansion can be computed before intermediate expressions in FORM or Mathematica become too large for the expansion to be completed. For the subsectors of I10, with each increase in the order of the Taylor expansion the intermediate expressions in the TayInt calculation increase in size by a factor of three. To put this into context, it is instructive to take a closer look at the first subsector of the integral I39,

$$\begin{aligned} \text{I39}_1 = \prod_{j=0}^{3} \int_0^1 dt_j \, \big(&(1 + t_0 + t_1 + t_2) \, (1 + t_0 + t_1 + t_2 + (1 + t_0)(t_1 + t_2) \, t_3) \\ &(1 + t_0 + t_1 + t_2 + (1 + t_0)(t_1 + t_2) \, t_3) \, m_1^2 - t_0 t_2 u - t_1 (s + t_0 m_2^2) \big)^{-1} \, . \end{aligned} \tag{9.21}$$

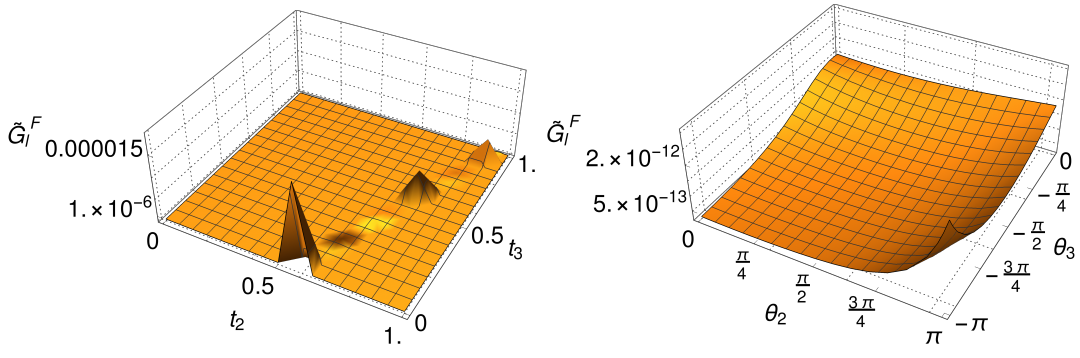
All subsectors of I39 have four Feynman parameters and four scales, s, u, m_1 and m_2. Because of the complexity of these sectors after performing the first integration exactly, an expansion beyond (typically) tenth order in the Taylor series is not possible beyond $\mathcal{O}(\epsilon^1)$ over threshold, due to the size of the algebraic expressions that are generated at intermediate stages.

OT5: Partitioning

Even if a higher-order Taylor expansion cannot be generated, the algebraic approximation for the Feynman integral can still be made as precise as required by using more partitions. It is important to notice that the increase in the number of partitions allows the circumvention of memory bottlenecks. The expressions for each partition can be truncated at a lower order in the Taylor series than the full expression, owing to the smaller distance from the expansion point. Increasing the number of partitions does increase the algebraic computation time required to obtain the series expansion, which can however be parallelised trivially. Once the result is computed, an instant evaluation at arbitrary phase space points is possible. Thus, an increased partitioning enables the result to meet a target precision. For example, the I10 sectors have three Feynman parameters and three scales and the Taylor series can be computed to beyond 10th order. Doubling the number of partitions at sixth order reduces the error in the real part by 91% and that of the imaginary part by 86%.

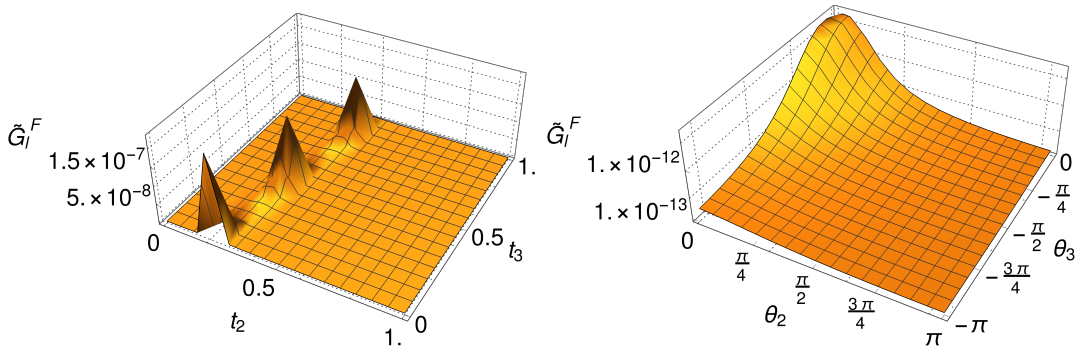
OT6: Taylor Expansion and Integration

The over-threshold part of the conceptual TayInt algorithm is implemented in each kinematic region that is over a mass threshold. To illustrate this in the case of multiple thresholds, the integrand of $\mathrm{I246}_1$ (not given here due to its length), Fig. 13.20(a), is plotted over the first threshold in Fig. 9.10 and over the second threshold in Fig. 9.11. Within each plot, the integrand is shown prior to running the conceptual TayInt algorithm on the left and after using the conceptual TayInt algorithm to determine the contour configuration on the right. In Fig. 9.10(a), the integrand simply in terms of the Feynman parameters contains threshold singularities but the complex integrand in Fig. 9.10(b) manifests smooth behaviour throughout the integration region. Likewise, in Fig. 9.11(a), the integrand simply in terms of the Feynman parameters contains more threshold singularities but the complex integrand in Fig. 9.11(b) still manifests smooth behaviour throughout the integration region. Thus, in both over-threshold regions, the conceptual TayInt algorithm can take integrands with threshold singularities and convert them into smooth integrands. These can then be calculated algebraically by means of a Taylor expansion to produce systematic approximations valid everywhere in each threshold region, in step OT6.



(a) (b)

Figure 9.10: A slice of the absolute value of the I246$_1$ integrand at $\mathcal{O}(\epsilon^0)$ in the first over-threshold region (a) without a complex mapping, $t_0 = 1$, $t_1 = \frac{1}{10}$, $t_4 = \frac{1}{10}$, $t_5 = 0$ and (b) with a complex mapping, $\theta_0 = -\pi$, $\theta_1 = -\frac{\pi}{10}$, $\theta_4 = -\frac{\pi}{10}$, $\theta_5 = 0$ and the contour orientation determined via the conceptual TayInt algorithm, setting $u = 3.2\,m_1^2$, $m_2 = \frac{m_1}{\sqrt{2}}$ and $m_1 = 173$ GeV.



(a) (b)

Figure 9.11: A slice of the absolute value of the I246$_1$ integrand at $\mathcal{O}(\epsilon^0)$ in the second over-threshold region (a) without a complex mapping, $t_0 = 1$, $t_1 = \frac{1}{10}$, $t_4 = \frac{1}{10}$, $t_5 = 0$ and (b) with a complex mapping, $\theta_0 = -\pi$, $\theta_1 = -\frac{\pi}{10}$, $\theta_4 = -\frac{\pi}{10}$, $\theta_5 = 0$ and the contour orientation determined via the conceptual TayInt algorithm, setting $u = 7.2\,m_1^2$, $m_2 = \frac{m_1}{\sqrt{2}}$ and $m_1 = 173$ GeV.

However, although the conceptual TayInt algorithm can handle multi-threshold integrals, the positions of the kinematic thresholds of each Feynman integral had to be entered by the user. In order to make the final TayInt algorithm fully self-contained, the ability to compute the locations of the thresholds was added to the conceptual TayInt algorithm, as will be explained in the next chapter.

10 | The TayInt Threshold Finder

10.1 Introduction

As the scope of the calculations being performed using the Standard Model increases, Feynman integrals with large numbers of internal massive lines must be evaluated. However, such integrals contain many singularities depending on the kinematic scales involved in the denominator of the integrand. These singular points are known as thresholds and are located according to the Landau equations [112]. As was shown in Chapter 9, the TayInt program calculates a result for Feynman integrals as a function of their kinematic scales. As the threshold singularities depend on these scales, this result is really a library of algebraic approximations, one for each over-threshold region. Thus, to produce an algebraic approximation for a Feynman integral with TayInt requires knowledge of the thresholds of the integral. Unfortunately, solving the Landau equations at the two-loop level cannot be done in full generality at present. Hence understanding this problem and finding new ways to obtain results for the Landau equations is subject to ongoing research, [128], [128], [129], [130]. Therefore, an original numerical method based on a Taylor expansion to approximately isolate the positions of the thresholds of Feynman integrals was written and incorporated into the TayInt algorithm, although it cannot distinguish real thresholds and pseudo thresholds. This is subsequently discussed fully herein.

10.2 Overview

In order to develop the conceptual TayInt algorithm into the final TayInt algorithm, the ability to locate the thresholds of the input Feynman integral was added. This will be step U2 of the final TayInt algorithm (as will be demonstrated in Chapter 13 the TayInt algorithm can compute algebraic approximations for divergent integrals so step U1 of the conceptual TayInt algorithm is dropped from the final algorithm). In the presence of a threshold a Taylor expansion breaks down by construction. This was the Minkowski problem that lead to the development of the over-threshold part of conceptual TayInt algorithm. In order to make this algorithm fully autonomous the thresholds are identified by constructing ratios of different orders in a Taylor expansion of the Feynman integral and selecting the kinematic points at which the ratios become uncontrollably large.

This is accomplished by first assembling a list of all the potential thresholds in the input Feynman integral. Next, the positions in that list of the kinematic points which lead to large ratios of Taylor expansion orders are selected. All possible ratios using a Taylor expansion up to sixth order are constructed for two sets of kinematic points, ensuring that no thresholds are missed. This is because the aim of the threshold-finding algorithm is to locate all the possible thresholds of a Feynman integral. Failing to identify a true threshold would lead to the final TayInt approximation of the integral breaking down at certain kinematic points. But, misidentifying a kinematic point which is not a threshold as a true threshold will only lead to the TayInt algorithm running for longer, as it will be implemented in more kinematic regions. Thus, the threshold-finding algorithm attempts to ensure that all true thresholds are found so that the TayInt approximation of the Feynman integral is accurate at all kinematic points, accepting the risk of increasing the number of times the algorithm has to be run. However, even though not all the kinematic points selected for use as thresholds by TayInt may actually be thresholds, by construction they are still points at which a Taylor expansion struggles. The more difficult it is to approximate the subsectors of the Feynman integral with a Taylor expansion, the more difficult it is to choose the contours and partition sets which allow accurate approximations to be generated using it. Therefore, running the TayInt algorithm within the regions intermediate to those points at which a Taylor expansion endures sustained breakdown (true thresholds or not) eases the process of selecting the optimal contour configurations to represent each subsector of the Feynman integral and the partition set with which to calculate it. This is especially true for Feynman integrals whose subsectors contain many turning points, which are difficult to describe using a Taylor expansion.

To achieve this goal, the threshold-finding algorithm adheres to the four principles of the final TayInt algorithm (which will be used again in the over-threshold part of the final TayInt algorithm, described in Chapter 11), which are:

1. P1, Mimicry: imitate the actual calculation as closely as possible. As the TayInt calculation is carried out using a Taylor expansion, the threshold are searched for by using ratios of orders in a Taylor expansion.

2. P2, Maximise information: because most of the possible mass threshold are not true thresholds of the Feynman integral, in order to correctly locate those that are true thresholds, the threshold-finding algorithm needs to be as informed as possible and check all the possible ratios of orders in a Taylor expansion up to sixth order using two sets of kinematic points. Sixth order is chosen as a compromise between being sufficiently exhaustive and keeping the run time reasonable.

3. P3, Negative exclusion: the threshold-finding algorithm cannot exactly locate the thresholds, as this is an as-yet unsolved problem in theoretical physics. So it excludes those mass thresholds which are clearly not causing the Taylor expansion to break down. This leaves those ratios indicative of breakdown which cluster around certain kinematic points. This is taken as an indication of the presence of a threshold in that kinematic vicinity.

4. P4, Safety first: As the accuracy and kinematic generalisability of the final TayInt result for a Feynman integral are more important than the run time of the algorithm, the threshold-finding algorithm makes sure that it has found all the true thresholds, at the risk of including a few pseudo thresholds.

By adhering to these principles, the location of the thresholds of the input Feynman integral helps to achieve the three objectives of TayInt:

1. O1: Accuracy. By finding the thresholds of the Feynman integral the final TayInt algorithm can then determine the contour configurations that lead to accurate numerical approximations when arbitrary kinematic values are inserted.

2. O2: Improvement. By finding the thresholds of the Feynman integral the final TayInt algorithm can also find a partitioning of the subsector integrands that improves the accuracy or precision or both of the approximation.

3. O3: Kinematic generalisability. Locating the thresholds of the Feynman integral ensures that the algebraic library of approximations evaluates to accurate and precise numerical approximations at any kinematic point.

which to be fully achieved require the development of the over-threshold part of the algorithm, as will be explained in Chapter 11.

10.3 Illustration

The previously stated principles (P1-4) lead to the threshold-finding algorithm having multiple steps. This is illustrated in the flow chart, given in Fig. 10.1. In the flow chart, after the initial input and setup in step T1, the algorithm proceeds from left to right through and downwards within, each layer. It then combines the results of the two layers to generate the final list of thresholds. A glossary containing definitions of all the important elements of the threshold-finding algorithm is given in Appendix A.

10.4 Quantitative Analysis

In order to carry out the steps illustrated above the concepts underlying them must be quantified. This will allow the thresholds of the input Feynman integral to be located in an algorithmic manner. Before the quantitative analysis begins, the ϵ^0 subsectors are inserted and the list of Feynman parameters in the integral is extracted from them. Only the ϵ^0 subsectors are required to locate the thresholds to all ϵ-orders.

10.4.1 T1: Generate the Lists of Potential Thresholds and the Kinematic Scanning Sets

The first quantitative method in step T1 is to generate the list of potential thresholds for the input Feynman integral. The lists the threshold-finding algorithm requires to select

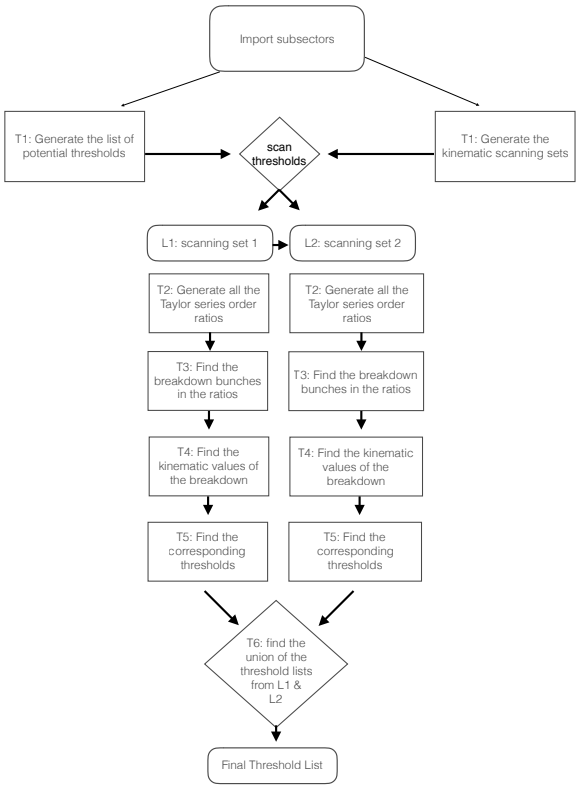


Figure 10.1: The relationship between the steps in the threshold-finding part of the final TayInt algorithm, step U2.

the relevant thresholds are also generated. First, the numerical list of distinct values for each mass in the Feynman integral is inserted from the user input file. This is a list of numbers, such as `KineMassesVal={173}` in the case of an equal-mass integral. The list of symbolic masses for each propagator, `kinemass` is also generated from the input provided by the user and the null entries are deleted, leading to `kinemassNZ`. A list of all its possible subsets, is then generated using the code:

```
kinemassNZCombos=Subsets[kinemassNZ,#]
                    &/@ Range[1,Length[kinemassNZ]] .
```
(10.1)

In order to find every possible combination of masses, a list of all possible tuples of 1 and -1 with length ranging from one to the number of non-trivial masses is generated with the code:

```
SignMass=Tuples[{+1,-1},#]&/@ Range[1,Length[kinemassNZ]] .
```
(10.2)

A list of all the possible thresholds is generated by multiplying together the `kinemassNZCombos` and `SignMass` lists and squaring and summing each resultant list. The `SignMass` list ensures the inclusion of every possible combination of signs per number of masses, thus guaranteeing that every possible way that the propagators could contribute to a threshold is included. The numerical values for the masses are then inserted and any duplicates are deleted. This is performed using the code:

```
ThreshCand =Sort[DeleteDuplicates[Flatten[Table[Table[
(Total[kinemassNZCombos[[i]][[#]]*SignMass[[i]][[j]]])^2
&/@ Range[1,Length[kinemassNZCombos[[i]]]],
{j,1,Length[SignMass[[i]]]}],
{i,1,Length[kinemassNZCombos]}]]]
/.Thread[DeleteDuplicates[kinemassNZ] → KineMassesVal]] .          (10.3)
```

The input Feynman integral must then be scanned to determine which of the candidate thresholds in `ThreshCand` are selected as thresholds by the algorithm. In order to do this, a list of numerical kinematic values must be inserted and the subsequent behaviour of the Feynman integral analysed. To this end, in accordance with the principle of maximising information, two scanning lists of values for the kinematic scales are generated. Firstly, `ScanList`, in which all the scales (each slightly offset, to avoid unphysical configurations) are varied. Secondly, `ScanListb`, in which only the scale that the user wishes to vary in the numerical TayInt approximations is allowed to run. The scales are varied in units of $0.25 \cdot \sum_i \texttt{KineMassesVal}_i^2$, where i labels the entries in `KineMassesVal`. This is to account for all possible mass contributions from -1 times this unit to whichever is higher of eleven times this unit or the maximum kinematic value at which the user desires algebraic approximations. At the end of step T1, the relevant user input has been imported and the lists of potential thresholds and the kinematic scanning sets which will be used to assess them generated.

10.4.2 T2: Generate All the Taylor Series Order Ratios

In step T2, the ratios must be generated that will be used to search for the locations of the Feynman integral's thresholds. Using the TayInt calculation code (see Appendix B.1) in adherence with the mimicry principle (P1), the input Feynman integral is therefore calculated in full up to sixth order in the Taylor expansion. In accordance with the principle of maximising information (P2), all possible ratios, namely:

$$\frac{\texttt{TayList4}}{\texttt{TayList0}}, \frac{\texttt{TayList2}}{\texttt{TayList0}}, \frac{\texttt{TayList4}}{\texttt{TayList2}}, \frac{\texttt{TayList6}}{\texttt{TayList0}}, \frac{\texttt{TayList6}}{\texttt{TayList2}}, \frac{\texttt{TayList6}}{\texttt{TayList4}}, \tag{10.4}$$

and;

$$\frac{\texttt{TayList4b}}{\texttt{TayList0b}}, \frac{\texttt{TayList2b}}{\texttt{TayList0b}}, \frac{\texttt{TayList4b}}{\texttt{TayList2b}}, \frac{\texttt{TayList6b}}{\texttt{TayList0b}}, \frac{\texttt{TayList6b}}{\texttt{TayList2b}}, \frac{\texttt{TayList6b}}{\texttt{TayList4b}}, \tag{10.5}$$

are computed, where the number denotes the order of the Taylor coefficient and the letter `b` denotes that the numerical values from the second list, `ScanListb`, were used to produce the results.

So, for example, `TayList4` is the list of results obtained by taking the fourth order contribution to the Taylor expanded and integrated result for the Feynman integral and evaluating them at the kinematic points in `ScanList`. All of these lists of ratios are then grouped into two lists of lists, one corresponding to `ScanList` and one to `ScanListb`. These are called `TayListRat` and `TayListRatb` respectively:

```
TayListRat = {TayListRat40,TayListRat20,
              TayListRat42,TayListRat60,TayListRat62,
              TayListRat62,TayListRat64},                    (10.6)
```

```
TayListRatb = {TayListRat40b,TayListRat20b,
               TayListRat42b,TayListRat60b,TayListRat62b,
               TayListRat62b,TayListRat64b} .                (10.7)
```

At the end of the step T2 of the threshold-finding algorithm, two nested lists have been generated. These contain all of the information necessary to choose the thresholds from `ThreshCand`. These will be used to determine within which kinematic regions the over-threshold part of the final TayInt algorithm should be run.

10.4.3 L1 (T3): Find the Breakdown Bunches in the Order Ratios

Now that the two lists of ratios have been generated, they must be used. Firstly, certain criteria for sustained breakdown of the orders of the Taylor expansion must be given. Then, in the step T3 the kinematic regions must be identified in which the ratios satisfy these criteria. This is indicative of a threshold. From now on until step OT6, the same procedures are performed for both `TayListRat` and `TayListRatb`, in layers L1 and L2. For brevity, only the operations based on `TayListRat` are described below. Ratios at consecutive kinematic points meeting the sustained breakdown criteria are termed breakdown bunches. The criteria for breakdown are:

1. The ratio is greater than 1,

2. And greater than the ratio before it but smaller than the ratio after it, i.e, those points at which the ratio is large and increasing. The ratio being large indicates that the Taylor expansion is breaking down at that particular kinematic configuration; the ratio increasing indicates that this corresponds to a region in which the Taylor expansion breaks down by construction rather than an anomaly.

The positions of breakdown in each list of ratios are then stored in a nested list:

```
BunchPos = Table[(((((If[TayListRat[[i]][[#]] >1
&& TayListRat[[i]][[#]]>TayListRat[[i]][[#-1]]
&& TayListRat[[i]][[#+1]]>TayListRat[[i]][[#-1]],
Position[TayListRat[[i]],TayListRat[[i]][[#]]][[1]][[1]]])))
&/@Range[1,Length[TayListRat[[i]]]-1])
//.{Indeterminate → 10000000})
//.{Null → Nothing},
{i,1,Length[TayListRat]}] ,                                    (10.8)
```

wherein the genuine infinities are replaced by very large numbers and the `Null` elements signifying that the breakdown criteria are not met are removed. The infinities are replaced by very large numbers because these are also indicative of the breakdown of the Taylor expansion. But, if they were left as `Indeterminate` they would not be identified as such.

The consecutive kinematic positions indicative of sustained breakdown in `BunchPos` are then separated into bunches, for each ratio in `TayListRat`, using the code:

```
SepBunch = Table[If[Length[BunchPos[[i]]]>0,
        If[BunchPos[[i]][[#+1]]==(BunchPos[[i]][[#]]+1),
        BunchPos[[i]][[#]]]
        &/@ Range[1,Length[BunchPos[[i]]]-1]],
        {i,1,Length[TayListRat]}] ,                            (10.9)
```

from which, at the end of step T3, generates a nested list of numbers denoting the positions in `ScanList` of the ratios at which sustained breakdown of the Taylor expansion of the Feynman integral occurs, for each ratio in `TayListRat`. This list is termed `ThreshBunches`.

10.4.4 L1 (T4): Find the Kinematic Values of the Breakdown

Next, in step T4, the kinematic values which generated the ratios in `ThreshBunches` are located, performed quantitatively as:

```
ThreshFac = Table[If[Length[BunchPos[[i]]]>0,
         Part[Flatten[ScanList[[#]][[1]]
         &/@Range[1,Length[ScanList]]],
         ThreshBunches[[i]][[#]]]
         &/@ Range[1,Length[ThreshBunches[[i]]]]],
         {i,1,Length[TayListRat]}] .                           (10.10)
```

10.4.5 L1 (T5): Find the Corresponding Thresholds

In step T5, the absolute value of the difference between those values in `ThreshFac` and the potential thresholds in `ThreshCand` is computed for each entry in `ThreshFac`, leading to a list of such differences for each entry of `ThreshCand`. The minimum value in each sub-list is then taken, resulting in a number quantifying the minimum difference for each potential threshold. This list is quantitatively assembled using the code:

```
ThreshFacDispl = Table[If[Length[BunchPos[[i]]]>0,
                 (Min[Abs[Flatten[ThreshFac[[i]]]-ThreshCand[[#]]]])
                 & /@ Range[1,Length[ThreshCand]]],
                 {i,1,Length[TayListRat]}] .                    (10.11)
```

If this minimum difference is below $\sum_i \texttt{KineMassesVal}_i^2$, then the corresponding potential threshold is selected from `ThreshCand`. This quantitative analysis is carried out using the code:

```
ThreshPos = Table[If[Length[BunchPos[[i]]]>0,
          If[Round[ThreshFacDispl][[i]][[#]]<=Total[KineMassesVal]^2,
          ThreshCand[[#]],Nothing]
          &/@ Range[1,Length[ThreshCand]]],
          {i,1,Length[TayListRat]}] ,                           (10.12)
```

which generates a list of kinematic values at which sustained breakdown of the Taylor expansion occurred, for each ratio in `TayListRat`.

10.4.6 L2 (T3-5): Overview

The steps from finding `BunchPos` to `ThreshFacDispl` (T3-5) are then repeated for the second list of kinematic configurations, `ScanListb`, generated in step T2, leading to a second list of kinematic values at which a sustained breakdown of the Taylor expansion occurred, `ThreshPosb`.

10.4.7 T6: Find the Union of the Threshold Lists from L1 & L2

The potential thresholds from the list `ThreshCand` are selected using each list of possible ratios based on a sixth-order Taylor expansion, inserting the kinematic values from the two scanning sets. The union of all the sublists within `ThreshPos` and `ThreshPosb` is then computed to find the final list of thresholds obtained, quantitatively performed as:

```
ThreshPosFin = Sort[DeleteCases[DeleteDuplicates[
             Join[Flatten[ThreshPos],Flatten[ThreshPosb]]],
             Null]] .                                           (10.13)
```

10.4.8 Summary:T1-T6

The advantage of this method is that it is very inclusive. It adheres to the principle of maximising information (P2) by incorporating information from different Taylor-generated ratios and different sets of kinematics. This ensures that all the thresholds of a Feynman integral are found. The cost of this is that some kinematic points are selected which are pseudo thresholds or not genuine thresholds. This means that the over-threshold part of the final TayInt algorithm will be run for more regions than is strictly necessary. However this will only increase the run time. Thus, the safety-first approach (P4) is preferred to missing some genuine thresholds and producing inaccurate approximations more quickly. All of the steps and the selection criteria employed within them are summarised in Tables 10.1 and 10.2 below.

Table 10.1: Summary of the individual sub-steps within the threshold-finding algorithm, which is step U2 of the final TayInt algorithm.

T1: import the subsectors, generate the list of potential thresholds and the kinematic scanning lists	
L1: scanning set 1	L2: scanning set 2
T2: Generate all the Taylor series order ratios	T2: Generate all the Taylor series order ratios
T3: Find the breakdown bunches in the order ratios	T3: Find the breakdown bunches in the order ratios
T4: Find the kinematic values at which sustained breakdown occurs in each order ratio	T4: Find the kinematic values at which sustained breakdown occurs in each order ratio
T5: Find the corresponding thresholds	T5: Find the corresponding thresholds
T6: Find the union of the list of thresholds selected in L1 and L2	

Table 10.2: Summary of the selection criteria used in the quantitative analysis of the threshold-finding part of the final TayInt algorithm. The grey rows denote the layer L2, in which the integral is analysed using the second kinematic scanning set. The index `i` runs over the number of ratios in `TayListRat` and `TayListRatb`, the index `#` runs over the number of kinematic points in `ScanList` and `ScanListb` at which the ratios are generated.

Step	Selection Criteria
	L1
T3	[(`TayListRat[[i]][[#]] >1`) && (`TayListRat[[i]][[#]]>TayListRat[[i]][[#-1]]`) && (`TayListRat[[i]][[#+1]]>TayListRat[[i]][[#-1]]`)]
T5	`ThreshFacDispl[[i]][[#]]`$< \sum_i$ `KineMassesVal`$_i^2$
	L2
T3	[(`TayListRat[[i]][[#]] >1`)&& (`TayListRat[[i]][[#]]>TayListRat[[i]][[#-1]]`)&& (`TayListRat[[i]][[#+1]]>TayListRat[[i]][[#-1]]`)]
T5	`ThreshFacDispl[[i]][[#]]`$< \sum_i$ `KineMassesVal`$_i^2$
	L1 $\cap$ **L2**
T6	`ThreshPosFin=` {`ThreshPos` $\cap$ `ThreshPosb`}

10.5 Application

The quantitative skeleton behind the illustration of the threshold-finding algorithm is now set in motion by its application to the non-planar I246 integral.

10.5.1 T1: Generate the Lists of Potential Thresholds and the Kinematic Scanning Sets

The first part of step T1 is to set up the list of potential thresholds, which requires assembling some kinematic information about I246 from the input provied by the user. In this case, the numerical list of distinct masses reads `KineMassesVal={173}` as there is only one mass. But, the threshold-finding algorithm generalises to multi-mass cases. As stated in the section on quantitative analysis, the list of masses for each propagator, in this case `{m,m,m,m,0,0}`, is used to generate a list of all its possible subsets excluding zero, namely:

Example 10.5.1

```
kinemassNZCombos={{{{m},{m},{m},{m}},
                  {{m,m},{m,m},{m,m},{m,m}, {m,m},{m,m}},
                  {{m,m,m},{m,m,m}, {m,m,m},{m,m,m}},
                  {{m,m,m,m}}} .                                    (10.14)
```

In order to assemble all the possible combinations of these masses, a list of all possible signs for each subset in `kinemassNZCombos` is also generated, which reads:

Example 10.5.2

```
SignMass={{{{1},{-1}},{{1,1},{1,-1}, {-1,1},{-1,-1}},              (10.15)
         {{1,1,1},{1,1,-1},{1,-1,1},{1,-1,-1},
         {-1,1,1},{-1,1,-1},{-1,-1,1},{-1,-1,-1}},
         {{1,1,1,1},{1,1,1,-1},{1,1,-1,1},{1,1,-1,-1},
         {1,-1,1,1},{1,-1,1,-1}, {1,-1,-1,1},{1,-1,-1,-1},
         {-1,1,1,1},{-1,1,1,-1},{-1,1,-1,1},
         {-1,1,-1,-1},{-1,-1,1,1},{-1,-1,1,-1},
         {-1,-1,-1,1},{-1,-1,-1,-1}}},
```

in the case of I246 and the list of potential thresholds reads;

Example 10.5.3

```
ThreshCand={0,29929,119716,269361,478864}                           (10.16)
```

after inserting $m = 173$ GeV^2, as described in the quantitative analysis subsection. In this case all the masses are equal. Nevertheless, the TayInt method generalises to situations where the masses are all different and still works equally well.

In order to generate numerical ratios to select those entries in `ThreshCand` which correspond to thresholds of I246, two sets of kinematic values must be used. As described in the quantitative analysis subsection, `ScanList` varies all the kinematic scales, whereas in `ScanListb` only the scale the user wishes to vary changes value. In the case of I246, the sensible scale to vary is u, so these scanning lists read:

Example 10.5.4

```
ScanList={{-29929.,-29929.,-29929.,29929},
          {-22446.7,-22446.7,-22446.8,29929},
          {-14964.5,-14964.5,-14964.5,29929},...,
          {314255.,314255.,314255.,29929},
          {321737.,321737.,321737.,29929},
          {329219.,329219.,329219.,29929}},                    (10.17)
```

Example 10.5.5

```
ThresListb={{-29929.,14964.5,-59858,29929},
            {-22446.8,14964.5,-59858,29929},
            {-14964.5, 14964.5,-59858,29929},...,
            {314255.,14964.5,-59858,29929},
            {321737.,14964.5,-59858,29929},
            {329219.,14964.5,-59858,29929}} .                  (10.18)
```

10.5.2 T2: Generate All the Taylor Series Order Ratios

In step T2, the ratios of a sixth-order integrated Taylor expansion over both kinematic scanning sets are calculated. However, it is not instructive to list these long lists of numbers, so no description of the details of this step in the case of I246 is needed.

10.5.3 L1 (T3): Find the Breakdown Bunches in the Order Ratios

The various ratios of the sixth-order integrated Taylor expansion over both kinematic scanning sets have now been generated. Next, in step T3, the positions of those kinematic points in `ScanList` at which breakdown of the approximation of I246 occurs are located. This is a list of numbers for each ratio in `TayListRat`, that reads:

Example 10.5.6

```
BunchPos=
{{7,8,9,10,11,15,16,17,20,
21,24,25,28,29,33,34,35}, .....
.....{6,7,8,9,10,11,15,16,17,20,
21,24,25,28,29,33,34,35}} .
```
(10.19)

The bunches of consecutive numbers are then assigned to sub-sub-lists of their own, leading to:

Example 10.5.7

```
ThreshBunches={{{7,8,9,10},{15,16},{20},{24},{28},{33,34}},
               ...,{{6,7,8,9,10},{15,16},{20},
              {24},{28},{33,34}}} .
```
(10.20)

To visualise this classification, the `TayListRat40` ratios are plotted in Fig. 10.2. Some of the ratios are too large to be seen on the plot. Because some of the visible ratios are very large, the low and high u regions are not well resolved, hence the $u \in [0, 30000]$ GeV^2 is shown using an inset plot.

10.5.4 L1 (T4): Find the Kinematic Values of the Breakdown

The numbers in `ThreshBunches` must next be replaced by the corresponding numerical values in `ScanList`. As described in the quantitative analysis subsection, the minimum absolute difference between each value and each potential threshold is then computed. In the case of I246, the minimal absolute differences for each ratio obtained by inserting the values in `ScanList` are:

Example 10.5.8

```
ThreshFacDispl={{14964.6,0.05,7482.2,52375.7,261879.0},...,
               ,...,{7482.3,0.05,7482.2,52375.7,261879.0}},
```
(10.21)

which is the result of step T4.

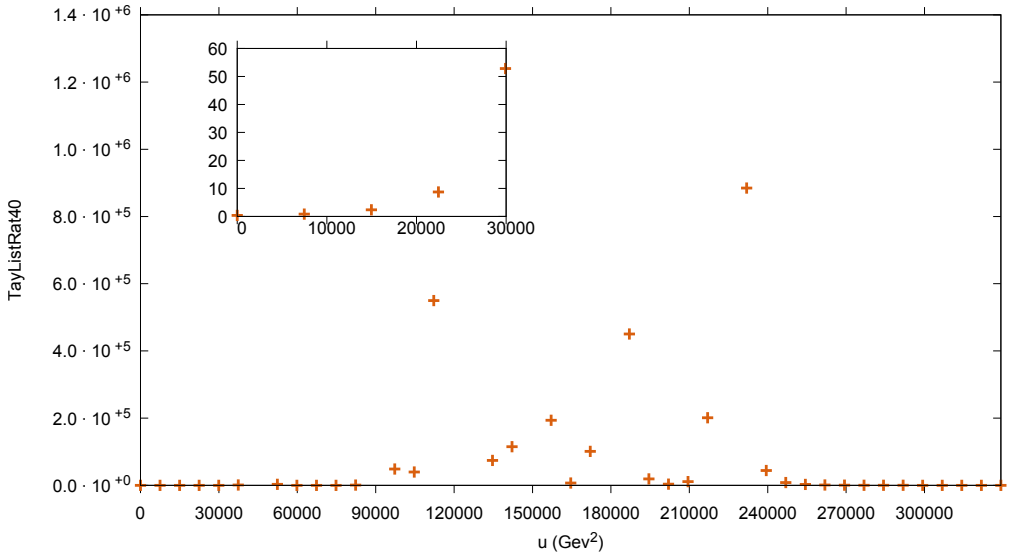


Figure 10.2: The ratio of the fourth-order and zeroth-order contributions to the TayInt result for the ϵ^0 coefficient of I246 (`TayListRat40`) plotted over the first range of u values, `ScanList`, used by the TayInt threshold-finding algorithm to locate the thresholds of the non-planar integral I246.

10.5.5 L1 (T5): Find the Corresponding Thresholds

In the case of I246, if any of these numbers is less than or equal to 14964.5, then in step T5 the corresponding threshold is selected from `ThreshCand`, which leads to:

Example 10.5.9

$$\begin{aligned}\texttt{ThreshPos=}&\{\{\{0,29929,119716\},\{0,29929,119716\},\{0,29929,119716\},\\ &\{0,29929,119716\},\{0,29929,119716\},\{0,29929,119716\},\\ &\{0,29929,119716\}\}\}\ .\end{aligned} \tag{10.22}$$

10.5.6 L2 (T3-5): Overview

The same procedure described above is also carried out for those ratios generated by inserting the numerical values in `ScanListb`. In the case of I246 this yields the list of thresholds:

Example 10.5.10

```
ThreshPosb={{{29929,119716,269361},{29929,119716,269361},
            {29929,119716,269361},{0,29929,119716,269361},
            {0,29929,119716,269361},{0,29929,119716,269361},
            {0,29929,119716,269361}}} .                    (10.23)
```

To visualise the classification of the threshold using `ScanListb`, the `TayListRat40b` ratios are plotted in Fig. 10.3. As before, some of the ratios are too large to be seen on the plot. Due to this, the low and high u regions are not well resolved. Thus the $u \in [0, 30000]$ GeV^2 region is shown using an inset plot.

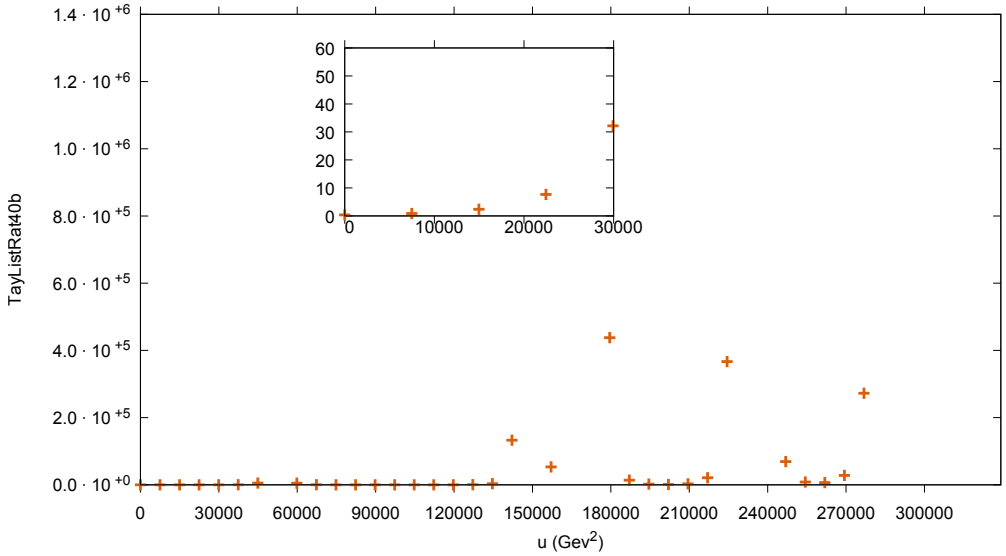


Figure 10.3: The ratio of the fourth order and zeroth order contributions to the TayInt result for the ϵ^0 coefficient of I246 (`TayListRat40b`) plotted over the second scanning set of u values, `ScanListb`, used by the TayInt threshold-finding algorithm to locate the thresholds of the non-planar integral I246.

10.5.7 T6: Find the Union of the Threshold Lists from L1 & L2

Therefore, following the principle of maximising information and the quantitative methods outlined in Table 10.2, the thresholds selected for use in the TayInt calculation of I246 are:

Example 10.5.11

$$\begin{aligned}\texttt{ThreshPosFin} &= \{\texttt{ThreshPos} \cap \texttt{ThreshPosb}\} \\ &= \{0, 29929, 119716, 269361\}\ \text{GeV}^2\ . \end{aligned} \tag{10.24}$$

10.5.8 Summary T1-T6

In the case of I246 only $\{0, 119716, 269361\}$ GeV^2 are true thresholds. However, the contours chosen in the $u \in [29929, 119716]$ GeV^2 region are only identical to those chosen in the $u \in [0, 29929]$ GeV^2 region at order ϵ^0. This repetition increases the run time of the final TayInt algorithm for this order. Nevertheless, the point $u = 29929$ is still one at which it is more difficult to approximate the integral I246 with a Taylor expansion. Thus, it is correspondingly more difficult to find a contour configuration that allows such an approximation to be accurate at that point. Hence, treating it as if it is a threshold still assists the subsequent over-threshold part of the final TayInt algorithm in choosing the optimal contours to achieve objective O1 and produce accurate approximations. This is shown by the over-threshold algorithm choosing different contour configurations and partition sets for I246 at orders ϵ^1 and ϵ^2. Moreover, the TayInt threshold location method is also selective. As can be seen, it does not just predict that thresholds are everywhere and it works for different kinematic scales and different topologies of Feynman graphs. For example, in the case of I10, I21 and the elliptic I59, the thresholds found by TayInt are $u = \{119716, 269361\}$ GeV^2, of which 119716 is a true threshold. In the case of I39 and the elliptic I59, the thresholds found by TayInt are $s = \{119716, 269361\}$ GeV^2, of which 119716 GeV^2 is a true threshold.

10.6 Recap

The TayInt threshold-finding algorithm aims to make the final algorithm more complete as the user no longer has to calculate and insert the values of the thresholds. The principles underlying it will next be applied to improve the over-threshold part of the TayInt algorithm, leading to the completion of its final form. The steps U2 and BT1-2 in the final TayInt algorithm have now been filled in (as they remain unchanged between the conceptual and final form), as shown by the boldface used for the corresponding labels in Table 10.3 below.

Table 10.3: Summary of the individual steps of the final TayInt algorithm, with the labels of the steps U1-U2 presented so far typeset in boldface.

U1: perform a sector decomposition on the finite integrals in the basis	
U2: locate all the thresholds of the Feynman integral	
Below threshold	Over threshold
BT1: $t_j \to y_j$	OT1: generate partition sets, full and partial contour configurations, kinematic training and cross-validation sets
BT2: Taylor expand the integrand and integrate	**OT2**: find partition:plain ratios for the full and partial contours
	OT3: Choose optimal full and partial contour by negative exclusion, backup with positive selection
	OT4: Perform kinematic cross validation
	OT5: Assess the optimal full and partial contours and choose between them
	OT6: Generate the uniform partition ratios on the chosen contours
	OT7: Assess the accuracy of the chosen contours to decide if a high-uniform partitioning can be used for all subsectors
	OT8: If not, use the accuracy and improvement assessment to decide if a low- or high-uniform partitioning is appropriate
	OT9: If not, analyse the subsectors on a per-variable basis and find the optimal varied partitioning
	OT10: Taylor expand and integrate

11 | The Final TayInt Algorithm

11.1 Introduction

In this chapter, the conceptual TayInt algorithm, now armed with the ability to automatically detect thresholds, will be extended so that it can produce algebraic approximations up to the level aimed for in the introduction, that of two-loop, four-point elliptic integrals.

11.2 Finalising the TayInt concept

The TayInt concept was successfully applied to two-loop three-point and four-point integrals with an internal mass scale that appear in the two-loop amplitudes for Higgs-plus-jet production. The central part of the concept that was crucial to this success was the over-threshold part of the algorithm introduced in Chapter 9. Using this, a contour configuration that led to an accurate approximation for each subsector of the integrals in over-threshold kinematic regions could always be found. A precise and accurate approximation was then generated by partitioning the surfaces of each subsector integrand into pieces, with the same number of pieces always being used for each variable of integration. Approximations were then produced for subsections of the region of integration and pieced together to obtain the full approximation. As each individual piece converges more quickly, the overall approximation obtained in this way is considerably more precise. To make this process more self-contained, an automated algorithm for locating the threshold of integrals was added to the TayInt algorithm in the previous chapter. Following this success, the next step was to apply the TayInt concept to elliptic integrals. But there are *three* key difficulties that arise when the TayInt concept is applied to integrals of this type, namely:

1. The proportion of contour configurations which lead to a representation of the subsector integrands that is suited for a Taylor expansion drops considerably.

2. The subsector integrand surfaces contain many more turning points and the variation between each subsector integrand increases.

3. The subsector integrands vary more as the kinematic scales are changed.

To address these issues, the following changes were made to the over-threshold part of the conceptual TayInt algorithm:

1. C1: the reduction in the number of viable contour configurations was addressed by considering contours in which only a subset of the Feynman parameters undergo a complex mapping and by making the method of selecting the contour configuration more rigorous. The first change increases the proportion of contour configurations suited to a Taylor expansion, the second ensures that the algorithm chooses one of them.

2. C2: the increased complexity of the subsector integrands is addressed by assessing whether or not a given subsector should be calculated with the same number of partitions in each integration variable. If not, the algorithm examines the behaviour of the integrand in each variable and tailors the number of partitions accordingly. This ensures that an accurate and precise approximation can be generated for even very complicated subsectors using TayInt.

3. C3: the greater variation exhibited by the subsectors as the kinematic scales are altered is addressed by using multiple kinematic samples within the algorithm. These are: a training set of kinematic points which encompass the largest possible region without crossing a threshold and three cross-validation sets which are subsets of this region. For a contour configuration to be chosen, it must display a sufficiently low generalisation error when applied to each of the cross-validation sets.

In the conceptual TayInt algorithm, one exact integration was performed on the subsectors if it could be completed within a certain time limit. However, this is no longer appropriate when more complicated Feynman integrals are considered. This is because performing one exact integration is both slow and unreliable. The once-integrated integrands often contain much more complicated mathematical structures than the original integrands, slowing down the subsequent algebra. Performing one exact integration is also not a reliable method of increasing the precision of the approximation. The amount of extra time that is required to do so does not reliably translate into a precision gain, as is the case when altering the number of partitions. Furthermore, for very complicated integrals, even one exact integration cannot usually be performed. Therefore, this part of the over-threshold algorithm is removed. TayInt can also produce approximations for divergent integrals (see Chapter 13). So, the process of generating a quasi-finite basis is also removed, which eliminates the external dependence on Reduze. The additions and deletions to the conceptual TayInt algorithm, coupled with the Threshold finder, lead to the final TayInt algorithm, summarised in Table 11.1. This method was then automated in Python (see Chapter 12).

11.3 Objectives

The final TayInt algorithm is set up to achieve the following objectives:

1. O1: Accuracy. Find a contour configuration that leads to an accurate numerical approximation independently of the kinematic values inserted.

2. O2: Improvement. Find a partitioning of the surfaces of the subsector integrands that improves the accuracy, precision (or both) of the approximation as necessary.

3. O3: Kinematic generalisability. Do so in an automated way that generalises to any ϵ order of any Feynman integral at any kinematic point.

11.4 Principles

To achieve these objectives, the final TayInt algorithm follows the principles listed below:

1. P1, Mimicry: imitate the actual calculation as closely as possible. The final TayInt algorithm must base its decision logic on maximising suitability for partitioning and minimising the kinematic generalisation error because the calculation is done using partitioned subsectors of the Feynman integral, which are subsequently evaluated at arbitrary kinematic points. As the precision of the approximation is controlled by using partitions rather than high orders in the Taylor expansion, the appropriate test for convergence is using ratios in approximations obtained using different partitions, not different orders.

2. P2, Maximise information: because the number of contour configurations that are suited to a Taylor expansion of the subsectors of a Feynman integrand are rare, the choice made by TayInt needs to be as informed as possible, eliminating as much chance as is feasible. To achieve this, fully complex and partially complex contour configurations are considered and repeatedly examined from multiple perspectives before a contour is ultimately chosen for each subsector.

3. P3, Negative exclusion: the TayInt approach searches for a representation of the subsectors of a Feynman integral that are globally accurate and precise enough, rather than those which are locally perfect. In order to encapsulate this principle in computer code, decisions are made by choosing those contour configurations which exclude any unsuitable features, rather than those which maximise suitable features.

4. P4, Safety first: only make decisions by a large margin. The principle here is always to maximise the accuracy and precision of the overall approximation for a Feynman integral, *not* the percentage of subsectors for which an optimally accurate and precise approximation is obtained, as one inaccurate approximation undermines the entire calculation. So cost function is the accuracy of the full TayInt approximation, not the percentage of subsectors for which an optimal approximation can be produced. For example, when considering very complicated integrals, it is possible to over-partition the surface of the integrand. If a subsector has an integrand that fluctuates considerably in a particular variable, then by using more

pieces the Taylor expansion will better represent it and generate a more precise approximation. However, if the subsector is flat in a given variable, then dividing the Taylor expansion into smaller pieces resolves detail that is not there, leading to an approximation that is less accurate. So it is better to choose a lower number of partitions and lose some precision in a few subsectors, than produce more precise approximations for those few but an inaccurate approximation for one. Therefore, the algorithm will only choose to use a high number of partitions for a subsector if it demonstrates its suitability for this by a large margin.

11.5 Illustration

The principles stated above lead to the final TayInt algorithm having multiple layers within its over-threshold part. This is illustrated in the flow chart, given in Fig. 11.1. In the flow chart, after the initial input and setup in step OT1, the algorithm proceeds from left to right through and downwards within, each layer.

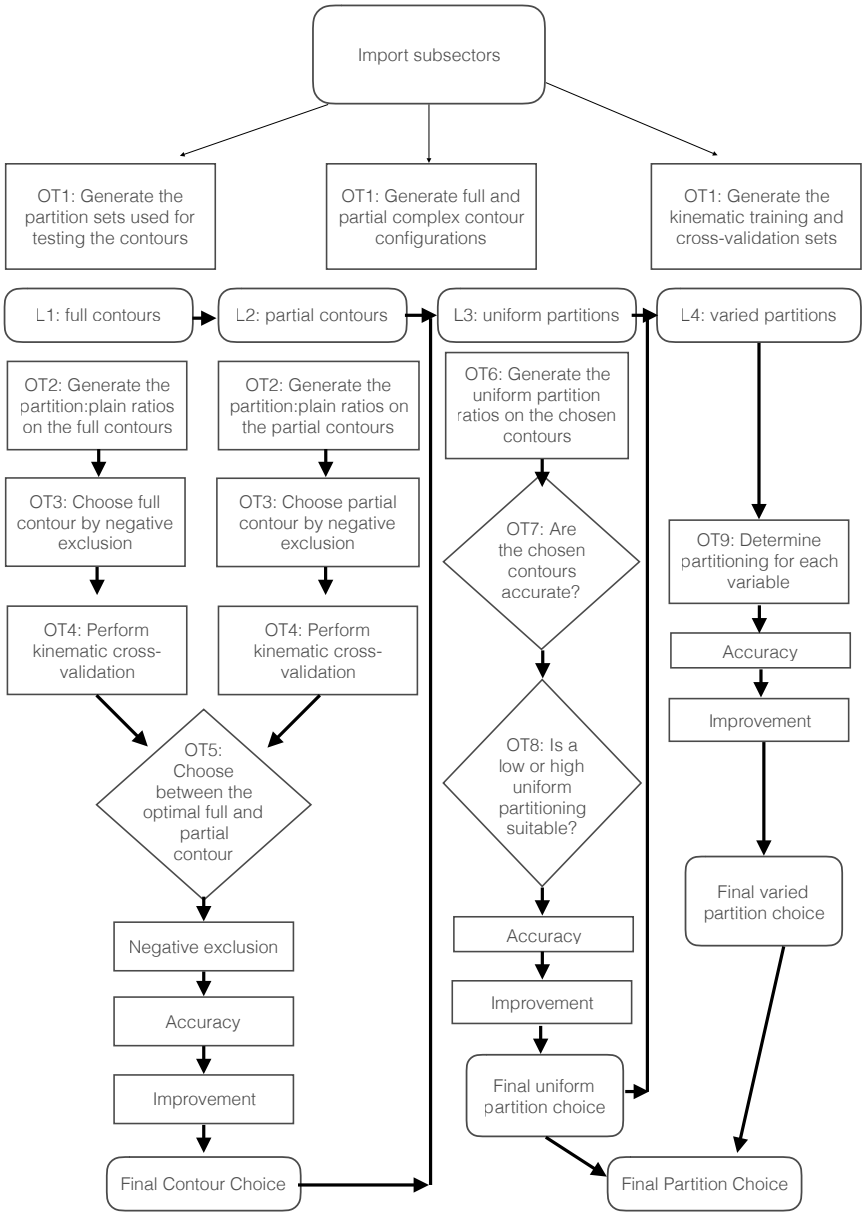


Figure 11.1: The relationship between the steps in the over-threshold part of the final TayInt algorithm.

11.6 Quantitative Analysis

The steps illustrated in Fig. 11.1 choose the contour configurations and partition sets that achieve the three objectives of the final TayInt algorithm (O1-3). In order to explain their application to for elliptic *and* the previously considered integrals in [1], the principles of the algorithm (P1-4) must be given a quantitative meaning using formulae. These formulae assess the characteristics of the different possible contour configurations and partition sets, for each subsector. They then produce numerical approximations to quantify terms such as accuracy, improvement, generalisable etc. The formulae that provide the numerical means of deciding how to choose the contour configuration and partition sets in each step of the over-threshold part of the algorithm will be defined below.

11.6.1 OT1: Generate the Partition Sets, Contour Configurations and Kinematic Training and Cross-Validation Sets

In this first step, the lists that the algorithm will work with are generated. Firstly, the potential full and partial contour configurations must be found, based on the number of Feynman parameters the subsectors have. To avoid the threshold singularities on the real contour of integration, the following mapping is performed within the subsectors: $t_j \to \frac{1}{2}-\frac{1}{2}e^{-i\theta_j}$, which opens up a variety of possible contours. This is a crucial step as it transforms the list of possible integration contours from $\{\{0,1\},\{0,1\},\{0,1\},\{0,1\},\{0,1\}\}$ to:

$$\begin{aligned}&\texttt{CCNIELcalc=}\{\{\{\texttt{0,}\pi\},\{\texttt{0,}\pi\},\{\texttt{0,}\pi\},\{\texttt{0,}\pi\},\{\texttt{0,}\pi\}\},\\&...\{\{\texttt{0,}-\pi\},\{\texttt{0,}\pi\},\{\texttt{0,}\pi\},\{\texttt{0,}\pi\},\{\texttt{0,}\pi\}\}\},\end{aligned} \tag{11.1}$$

in the case of an integral with five Feynman parameters after sector decomposition. Numerically, the mapping transforms the number of possible contours from one, to 2^j, where j is the number of Feynman parameters after sector decomposition. This list of possible contours, along with the list of transformed subsectors (`CAdomsec`), is created in step OT1. It means that there is now at least a possibility of avoiding the threshold singularities.

At the same time, the list of all the possible contours for which more than half of the Feynman parameters are mapped to the complex space is generated. This conforms to the principle of maximising information (P2). The list reads:

$$\texttt{SubFeynList2=}\{\{\theta_0,\theta_1,\theta_2,\theta_3,t_4,t_5\},...\{t_0,t_1,\theta_2,\theta_3,\theta_4,\theta_5\}\}, \tag{11.2}$$

in the case of an integral with five Feynman parameters after sector decomposition. Each of the subsectors is also transformed according to this list of partial mappings, generating a list (termed `CAdomsecSub`) of partially mapped subsectors with their Feynman-parameter dependence as specified in `SubFeynList2`. The corresponding list of partial

contours is also generated:

$$\begin{aligned}\texttt{CCNIELpartialcalc=}&\{\{\{\{0,-\pi\},\{0,-\pi\},\{0,-\pi\},\{0,1\},\{0,1\}\},\\ &\{\{0,-\pi\},\{0,-\pi\},\{0,\pi\},\{0,1\},\{0,1\}\},...\}\\ &...\{...\{\{0,1\},\{0,1\},\{0,-\pi\},\{0,\pi\},\{0,\pi\}\},\\ &\{\{0,1\},\{0,1\},\{0,-\pi\},\{0,-\pi\},\{0,\pi\}\}\}\}\ . \end{aligned} \tag{11.3}$$

Unlike `CCNIELcalc`, this is a nested list, because all the possible contour configurations must be found within each post-transformation variable constellation in `SubFeynList2`. The final TayInt algorithm will then analyse the full and partial contour representations of each subsector (OT2-4) and choose the optimal one accordingly (OT5). This choice is represented by a single number in the case of the full contours, denoting the position of that contour configuration in `CCNIELcalc`. In the partial case it is given by a bipartite list, denoting the position of the contour configuration within `CCNIELpartialcalc`. The first element denotes the transformation, the second the contour configuration within that transformation.

To make this assessment, the algorithm must have a way of numerically quantifying the suitability of a contour for calculating the subsector. The precision of the TayInt algebraic approximations are controlled by partitioning the subsector integrands. Hence, this quantification will be done by constructing ratios of the approximations obtained using a low number of partitions and those obtained using no partitions (plain). In order to do this, the low and plain partition sets, `NoPart` and `TestPart` must be generated. In the case of an integral with five Feynman parameters after sector decomposition, they read:

```
NoPart=ConstantArray[1,Length[varlist]]                    (11.4)
{{1},{1},{1},{1},{1}}
BaseList=
Prepend[ConstantArray[1, Length[varlist]-1],{1,1,1}]       (11.5)
{{1,1,1},{1},{1},{1},{1}}
TestPart=Permutations[BaseList]                            (11.6)
{{{1,1,1},{1},{1},{1},{1}},
{{{1},{1,1,1},{1},{1},{1}},
{{{1},{1},{1,1,1},{1},{1}},
{{{1},{1},{1},{1,1,1},{1}},
{{{1,1,1},{1},{1},{1},{1,1,1}}} .
```

However, without inserting kinematic points, these ratios will not be numbers. In order to use them to quantify the suitability of a contour for calculating a subsector, a numerical representation is mandatory. Therefore, in step OT1 the lists of kinematic points to be used by the algorithm must also be produced. According to the change C3 and to achieve objective O3 a training set and three cross-validation sets of kinematic points

must be generated. The training set of kinematic points begins with the threshold that bounds the first point at which the user desires results from below and ends with either the next threshold, or, if there are none, ten times the lowest threshold of the integral. This ensures that the training set used is as extensive as possible, to avoid the chosen contours being overfitted to a small region of kinematic points. However, once the optimal contours have been found using this full training set, they are cross-validated using three subsets of the training set (one low, one intermediate, one high), to ensure that they do indeed generalise to a set of kinematic points other than that with which they were chosen (objective O3). These sets are termed `TrainKine`, `CrossVal1Kine` (`[Min[TrainKine], 0.2·Max[TrainKine]]`), `CrossVal2Kine` (`[0.35·Max[TrainKine], 0.55·Max[TrainKine]]`) and `CrossVal3Kine` (`[0.75·Max[TrainKine], Max[TrainKine]]`). At the end of this step, all the subsidiary lists required for the algorithm to achieve objectives O1-3 are assembled.

11.6.2 L1 (OT2): Generate the Partition:Plain Ratios on the Full Contours

Steps OT2-4 search for the position of the contour in `CCNIELcalc` and `CCNIELpartialcalc` which is most suitable for achieving objectives O1 and O2 of the final TayInt algorithm. A suitable contour is defined as one that will lead to a calculation that achieves the objectives of TayInt:

1. O1: Accuracy. Those contours with partition:plain ratios in their variables which are close to 1. This means that the contour always produces a similar approximation regardless of the number of partitions used. Thus the choice of contour alone is enough to generate an accurate approximation for the subsector.

2. O2: Improvement. If the partition:plain ratios in the integration variables of a particular contour decrease quickly but not so quickly as to suggest the contour is inaccurate, then the accuracy and precision of the approximation improves substantially as more partitions are used. This is quantified by the partition:plain ratios lying in the range $[0.25, 1]$ as the number of partitions increases.

3. O3: Kinematic generalisability. If the mean absolute relative difference in the partition:plain ratios produced using different kinematic sets are within $0.5, 0.25$, and 0.5 for the three cross-validation sets, then the contour leads to accurate and precise approximations at arbitrary kinematic points.

In step OT2, the list of partition:plain ratios are constructed for each contour configuration of each subsector in `CCNIELcalc` using the lists generated in OT1, for both the full and partial contours. The formulae will only be given for the case of the full contours

here, as follows:

```
Table[TrainRatSensFracSec[k]=
(Mean[Abs[TrainGenPartSensKineSec[k][[#]]]] / Mean[Abs[TrainGenNoPartKineSec[k][[#]]]])
&/@ Range[1,Length[CCNIELcalc]],
{k,1,Length[SecList]}],                                                    (11.7)
```

where the Mean Absolute value is taken over the range of kinematic points in the kinematic training set. The index `k` runs over the subsectors and `#` denotes the different contour configurations. Step OT2 ends with the production of the list of partition:plain ratios using the kinematic training set, `TrainRatSensFracSec[k]`, adhering to the principle of mimicry (P1).

11.6.3 L1 (OT3): Choose the Full Contours via Negative Exclusion

The resulting `TrainRatSensFracSec[k]` is a list of ratios of length j, where j is the number of Feynman parameters in the integral after sector decomposition. Using these ratios, all those contours for which `TrainRatSensFracSec[k]` contains a value less than 0.5 or greater than 1.5 are removed. Then the optimal contour is selected from the entries of `CCNIELcalc` and `CCNIELpartialcalc` still remaining. This is using the principle of negative exclusion (P3), to find an accurate representation of the subsector. When none of the contour configurations exclude any unsuitable variables of integration (and so no accurate representation can be found) then a backup contour is selected. This backup contour must have the most integration variables that are improving quickly. "Improving quickly" is quantified by those ratios which fall in the range $[0.25, 1]$. This method is termed positive selection but the contour found in this way will only be used if none are found using negative exclusion. At the end of this step, in layer L1 (full contours), a simple list of numbers is generated, one for each subsector. The numbers denote the position of the optimal full contour in `CCNIELcalc`. At the end of step OT3 in layer L2 (partial contours), a list of bipartite lists is produced, denoting the position of the optimal partial contour in `CCNIELpartialcalc`. The lists are termed `TrainedContourSec[k]` and `PartialTrainedContourSec[k]` in layers L1 and L2, respectively. The production of these contour choices is in adherence with the principles of mimicry (P1) and negative exclusion (P3).

11.6.4 L1 (OT4): Perform Kinematic Cross Validation

The next quantitative assessment that must be made is to implement change C3 and achieve objective O3. This requires the mathematical construction of the partition:plain ratios as before. But, now this is done using each of the kinematic cross-validation sets,

as follows:

$$\texttt{Table[Cross1RatSensFracSec[k]=}\left(\frac{\texttt{Mean[Abs[Cross1GenPartSensKineSec[k][[\#]]]]}}{\texttt{Mean[Abs[Cross1GenNoPartKineSec[k][[\#]]]]}}\right)\texttt{\&/@ Range[1,Length[CCNIELcalc]],\{k,1,Length[SecList]\}]}\,, \tag{11.8}$$

where only one of the three lists of ratios is given, for brevity. The index `k` runs over the subsectors and `#` denotes the different contour configurations. This step is performed in the same way using the partial contours in layer L2. The relative difference between the ratios generated using the training set and those generated using each cross-validation set, are then calculated, as follows:

$$\texttt{ContCrossVal1=Mean[Transpose[}\frac{\texttt{Abs[Cross1RatSensFracSec[\#]-TrainRatSensFracSec[\#]]}}{\texttt{Abs[TrainRatSensFracSec[\#]]}}\texttt{]]\&/@ Range[1,Length[SecList]]}\,, \tag{11.9}$$

and so on for each cross-validation set. This produces a nested list. The inner list gives the cross-validation error for each contour, which is generated for each subsector.
Next, the contours within these nested lists which have such a relative difference within 0.5, 0.25, 0.5, in the three respective pairings for each subsector, are selected and their intersection computed. This leads to the list of contours `LowCrossVal[k]` for each subsector `k`. The difference in the acceptance threshold is because in the first and third cross-validation sets the approximations are naturally worse, since these sets contain kinematic points close to the threshold or very far above it. The Taylor expansion breaks down by construction in such regions, so a more generous margin is needed. If the previous optimal contour `TrainedContourSec[k]` (full, layer L1) or `PartialTrainedContourSec[k]` (partial, layer L2) intersects with one of the contours in `LowCrossVal[k]` (full, layer L1) or `PartialLowCrossVal[k]` (partial, layer L2), it is deemed valid. If it is *not*, then the entire procedure of OT2-3 is repeated but only for those contours in `LowCrossVal[k]`, which have a low generalisation error by construction. The lists of optimal contours at the end of this step, in the full and partial cases, are termed `ChosenContourSec[k]` and `PartialChosenContourSec[k]`. These two lists must now be compared to select the ultimate chosen contour for each subsector, `k`.

11.6.5 OT5: Choose Between the Optimal Full and Partial Contour

In order to implement change C2 and quantitatively compare the optimal full and partial contour configurations, the corresponding entries from the nested list of ratios must be selected and placed in another nested list. However, there are four possible lists of ratios that could be used for such a comparison, corresponding to each set of kinematic points generated in step OT1. To decide between them in an informed manner, the proximity of the optimal full and partial ratios generated using the kinematic training

set is numerically quantified. This quantification is done by calculating the fractional difference between the mean absolute value of the partition:plain ratios on the optimal full and partial contours:

```
FuParProx=(Abs[Mean[
TrainRatSensFracSec[#][[ChosenContourSec[#]]]]-
Mean[TrainPartialRatSensFracSec[#]
[[PartialChosenContourSec[#][[1]]]]
[[PartialChosenContourSec[#][[2]]]]]]/
(Max[{Abs[Mean[TrainRatSensFracSec[#][[ChosenContourSec[#]]]]],
Abs[Mean[TrainPartialRatSensFracSec[#]
[[PartialChosenContourSec[#][[1]]]]
[[PartialChosenContourSec[#][[2]]]]]}])
&/@Range[1,Length[SecList]] .                                    (11.10)
```

If this number, `FuParProx[[k]]` for subsector `k`, is greater than 0.15 then there is a clear difference between the optimal full and partial contour. This is enough to allow the TayInt algorithm to reliably ascertain which is the more suitable choice in the ensuing code. In that case, the nested list of full and partial partition:plain ratios is constructed using the ratios generated by the full training set of kinematic points. This reads:

```
TrainFuParComp={TrainRatSensFracSec[#][[ChosenContourSec[#]]],
PartialRatSensFracSec[#][[PartialChosenContourSec[#][[1]]]]
[[PartialChosenContourSec[#][[2]]]]} .                           (11.11)
```

If `FuParProx[[k]]` for subsector `k` falls below 0.15 then the margin between the optimal full and partial contours on the training set of kinematic points is not clear enough to allow the algorithm to be unaffected by the natural deterioration in the partition:plain ratios at the beginning and end of the training set. The ratios will naturally be poorer at these extremes, as close to threshold and very far above it the convergence of the Taylor expansion will start to break down and deteriorate by construction. Therefore, the partition:plain ratios will be larger at these kinematic points.

To elaborate, if `FuParProx[[k]]` for subsector `k` falls below 0.15, then the larger scale of the end point partition:plain ratios may mean that the superiority of a contour in the bulk of the training set goes unnoticed due to it having larger ratios at the beginning and end. For example, the optimal full contour may perform better over the bulk of the kinematic training set but worse at theend points, in which region both are poor in any case. However, because the ratios are so much larger at theend points, the mean over the kinematic training set is biased towards the partial contour, despite the fact that the full contour leads to the more accurate representation of the subsector integrand. This could lead to a contour being chosen from the optimal full and partial options that

generates less accurate approximations in the kinematic bulk but slightly more accurate ones at theend points. This is highly undesirable as it means that the contour that best minimises the extent of the breakdown of the approximation at the kinematic end points is chosen rather than the contour that maximises the accuracy and potential for improvement of the approximation.

Therefore, in this case, the list of partition:plain ratios is assembled using those ratios generated with the second cross-validation set of kinematic points. This is the middle of the training set and is thus free of any points at which the contours are expected to be naturally poorly suited for a Taylor expansion. Therefore, the differences in the bulk of the training set can be clearly seen by the algorithm. Then the contour is chosen accordingly. It is implemented using the equation:

```
CrossFuParComp={Cross2RatSensFracSec[#][[ChosenRatContourSec[#]]],
Cross2PartialRatSensFracSec[#][[PartialChosenRatContourSec[#][[1]]]]
[[PartialChosenRatContourSec[#][[2]]]]} .                    (11.12)
```

The final nested list of partition:plain ratios for the optimal full and partial contours is then assembled using an `if` statement,

```
FuParComp=If[FuParProx[[#]]>0.15,
TrainFuParComp[[#]],CrossFuParComp[[#]]]
&/@Range[1,Length[SecList]] ,                                (11.13)
```

and likewise for the real and imaginary parts of the ratios, generating the lists `FuParReComp` and `FuParImComp`. This robust approach to deciding between the optimal full and partial contours is part of the implementation of change C1. Continuing with that implementation, the list `FuParComp` is then scrutinised from three different quantitative perspectives to decide whether the optimal full or partial contour will be chosen for use in the TayInt calculation. These perspectives are those of negative exclusion, accuracy and improvement, described next.

Negative Exclusion

1. Firstly, if only one of the optimal contours contains no partition:plain ratios indicative of unsuitability for the TayInt calculation and the other contains more, it is immediately selected. Quantitatively, the full/partial contour is selected if it and only it, has all of its partition:plain ratios in `FuParComp` in the range $[0.5, 1.25]$. A narrower range is used than was employed when selecting the contours in OT2, because the two contours in this step have already been through a selection process, so a more stringent decision boundary needs to be used to separate them.

Accuracy

2. If the optimal full and partial contours cannot be separated by negative exclusion, the contour with the most accurate partition:plain ratios is selected. Accuracy is

quantified by those ratios in the range $[0.8, 1.2]$. The partition:plain ratios falling in this range indicate that the change of partition does not much affect the smoothness of the integrand. Hence, the representation of it is an accurate one and the partitioning is only needed to aid precision.

Improvement

1. At this point, the unsuitable contours have been avoided and the accurate ones have been selected, thus, in the remaining cases, it is less critical whether a full or a partial contour is chosen. In order to analyse their suitability more precisely, the difference between the number of integration variables which have partition:plain ratios in the range $[0.5, 0.7]$ (improvement) and those in the range $[0, 0.5] \cup [1.25, \infty)$ (unsuitable) is computed. The contour maximising this difference is selected. The bias towards the unsuitable ratios is deliberate, as accuracy is more important than improvement and a contour must only be selected on the basis of improvement if it is already sufficiently accurate. This logic adheres to the safety-first principle (P4) and makes the choice as robust as possible.

At the end of step OT5, the final list of contour choices, `CCFinal`, is assembled. This is a list of single numbers and two-fold lists because it is composed of full and partial contour configurations, in adherence with the principle of maximising information (P2).

11.6.6 L3 (OT6): Generate the Uniform Partition Ratios on the Chosen Contours

Now that the contour configurations have been chosen for each subsector, the next step, OT6, is to implement change C2 and decide which form of partitioning is suitable for producing an accurate, precise and general approximation (O1-3). There are three possibilities:

1. A low-uniform partitioning, in which all the integration variables are subject to the same low number of partitions, `{1,1,1}`;

2. A high-uniform partitioning, in which all the integration variables are subject to the same high number of partitions, `{1,1,1,1,1,1}`;

3. A varied partitioning, in which each integration variable is subject to a different number of partitions and a different partition set is produced for calculating the real and imaginary part of the subsector.

The first step in deciding the nature of the partitioning that is to be used in calculating each subsector, `SecList[k]`, is to calculate it on the chosen contour, `CCFinal[k]`. This is done using two uniform partitionings with a low number of partitions, `{1,1,1}` and `{1,1}` for every integration variable. Numerical approximations corresponding to these partition sets can be quickly produced over the training set of kinematic points, `TrainKine`. At the same time, numerical approximations using no partitions are generated. All

these numbers are stored in lists and termed `Hom3PartSec[k]`, `Hom2PartSec[k]` and `PlainPartSec[k]` for subsector `k`, respectively. The absolute value, real and imaginary parts of the ratios between the approximations produced with the uniform partitioning and without partitioning are then calculated, as follows:

```
Hom3PlainAbsRat=
 Mean[Abs[Hom3PartSec[#]]]
Mean[Abs[PlainPartSec[#]]]
&/@Range[1,Length[SecList]] ,                                    (11.14)
```

```
Hom3PlainReRat=
 Mean[Abs[Re[Hom3PartSec[#]]]]
Mean[Abs[Re[PlainPartSec[#]]]]
&/@Range[1,Length[SecList]] ,                                    (11.15)
```

```
Hom3PlainImRat=
 Mean[Abs[Im[Hom3PartSec[#]]]]
Mean[Abs[Im[PlainPartSec[#]]]]
&/@Range[1,Length[SecList]] ,                                    (11.16)
```

```
Hom2PlainAbsRat=
 Mean[Abs[Hom2PartSec[#]]]
Mean[Abs[PlainPartSec[#]]]
&/@Range[1,Length[SecList]] ,                                    (11.17)
```

```
Hom3Hom2AbsRat=
Mean[Abs[Hom3PartSec[#]]]
Mean[Abs[Hom2PartSec[#]]]
&/@Range[1,Length[SecList]] .                                    (11.18)
```

At the end of step OT6 the ratios which will be used to determine the form of the partitioning for those subsectors for which a uniform partition set is suitable have been generated and the next step is to use them.

11.6.7 L3 (OT7): Assess the Probable Accuracy of the Chosen Contours

Though this may seem superficially similar to the quantitative analysis that was performed to choose the contours in step OT2, it is in fact very different. This is because the contours being used in Eqs. (11.14)-(11.18) are the *chosen* contours, so it is expected that they yield accurate approximations. In the case of very complicated integrals however, it is usually always necessary to match the chosen contour with the optimal partition

set in order to generate a sufficiently accurate TayInt approximation. To assess how well the chosen contours are already performing without paying any attention to the partitioning, in step OT7 the mean of `Hom3Hom2AbsRat[k]` over the subsectors `k` is computed. If it is in the range $[0.95, 1.05]$ then the approximations produced using double and triple uniform partitions are very close to each other. Thus, the accuracy of the approximation is already guaranteed by the chosen contours and the uniform partitionings can be increased arbitrarily to raise the precision. If this criterion is met, then a six-fold uniform partitioning `{1,1,1,1,1,1}` is used for each sector and no further analysis is carried out. Note that, for the integrals I10 and I39, the final TayInt algorithm would always stop here.

11.6.8 L3 (OT8): Assess the Suitability of Low- or High-Uniform Partition Sets

When considering elliptic integrals however, it may be that the partition choice must be coupled with the contour choice to ensure an accurate representation of the subsector approximation (and the partitioning is not just needed as a tool to increase precision), implementing change C2. This can be the case if the subsectors are very complicated and contain many variables in which they behave very differently, with many turning points. This is identified by the the mean of `Hom3Hom2AbsRat[k]` over the subsectors `k` lying outside of the range $[0.9, 1.05]$. This shows that there is a risk of certain subsectors needing a particular partition set in order to be calculated accurately. If this is the case, then in step OT8 the final TayInt algorithm must decide whether a low-uniform, high-uniform, or varied partitioning is needed for each subsector, using the ratios generated in step OT6. This decision is made by using the criteria of accuracy and improvement.

Accuracy

To determine whether a low or a high-uniform partitioning is sufficient, the ratios in `Hom3PlainAbsRat` and `Hom2PlainAbsRat` as well as their real and imaginary counterparts, `Hom3PlainReRat` and `Hom3PlainImRat` are used. For those subsectors for which all four of these ratios fall within $[0.95, 1.05]$, the six-fold uniform partitioning is used. As noted in Subsection 11.6.5, this is because any changes due to increasing the partitioning are small and a matter of precision increase only. Therefore the contour produces accurate approximations even with a low number of partitions, so a high number of partitions can be uniformly applied.
In accordance with the safety-first principle (P4), this is an extremely conservative manner of opting for a high-uniform partitioning. However, if the mean of
`Hom3Hom2AbsRat[[k]]` over the subsectors `k` is not in the range $[0.95, 1.05]$, the precision of the approximations for the subsectors will not always increase as the number of partitions is increased. Therefore, rather than risk over-partitioning a subsector and losing accuracy in the approximation, the risk of losing some precision by occasionally under-partitioning a subsector is accepted instead.

Improvement

It is not just initial accuracy that qualifies a subsector as suitable for a high number of uniform partitions but also the extent to which the approximation *improves* by increasing the number of partitions uniformly. This is judged to be the case if either `Hom3PlainAbsRat[[k]]` or `Hom2PlainAbsRat[[k]]` falls between 0.4 and 0.6. The lower boundary is set at 0.4. This excludes those ratios which suggest a lack of initial accuracy (due to being too low), because of the partitioned approximation being much smaller than the plain approximation. Now that those subsectors suitable for a high-uniform partitioning have been identified, the next step is to check if any require a varied partitioning as well as a chosen contour in order to generate an accurate approximation.

Lack of Accuracy and Improvement

If *both* `Hom3PlainAbsRat[k]` and `Hom2PlainAbsRat[k]` exceed 1.1 then the uniform partitioning is leading to uncontrolled growth of the approximation and a different partition set needs to be chosen for each individual variable of integration.

Remainder

Those subsectors which do not fall into any of the above criteria are assigned a low-uniform partition set. In these cases, the approximation is improving with a uniform partition set but not quickly enough to warrant a high number of partitions. The logic used to choose the partition type for each subsector, k, is summarised in Eq. 11.19.

$$\texttt{kth partition Type=}\begin{cases}\texttt{High Homogeneous}, \\ \texttt{0.9<Mean[Hom3Hom2AbsRat[k]]<1.1} \\ \texttt{High Homogeneous}, \\ \texttt{0.9< Hom3PlainAbsRat[k]<1.1} \\ \texttt{\&\&,0.9<Hom2PlainAbsRat[k]<1.1} \\ \texttt{\&\&,0.9<Hom3PlainReRat[k]<1.1} \\ \texttt{\&\&,0.9<Hom3PlainImRat[k]<1.1} \\ \texttt{High Homogeneous}, \\ \texttt{0.4< Hom2PlainAbsRat[k]]<0.6} \\ \texttt{|| 0.4<Hom3PlainAbsRat[k]]<0.6} \\ \texttt{Precise, Hom2PlainAbsRat[k]]>1.1} \\ \texttt{|| Hom3PlainAbsRat[k]]>1.1} \\ \texttt{Low Homogeneous, otherwise}\end{cases} \tag{11.19}$$

At the end of step OT8, all those subsectors which can be accurately and precisely calculated using a uniform partition set have been assigned such a set.

11.6.9 L4 (OT9): Determine the Varied Partitioning for those Subsectors Requiring it

If a subsector on its chosen contour cannot be precisely represented by using a uniform partitioning, then this is because there are some variables with respect to which the subsector integrand is flat and unvarying, so using a uniform set of partitions does not fit its behaviour. Therefore, in step OT9, these variables need to be identified and not partitioned at all. In order to do this, partition sets of increasing fineness in which each variable is separately partitioned, are generated and stored in a list, called `ParTest`. For example, in the case of a Feynman integral with three Feynman parameters after sector decomposition, this reads:

```
ParTest=
{{{{1,1},{1},{1}},{{1},{1,1},{1}},{{1},{1},{1,1}}}
{{{1,1,1},{1},{1}},{{1},{1,1,1},{1}},{{1},{1},{1,1,1}}}
{{{1,1,1,1},{1},{1}},{{1},{1,1,1,1},{1}},{{1},{1},{1,1,1,1}}}} .
```
(11.20)

This generates a nested list with each variable partitioned using partition numbers from 2 to 4. Using the chosen contour, given in `CCFinal`, numerical approximations are generated using each of the partition sets in `ParTest`. This is done for each subsector which was found to be unsuited to a uniform partitioning in OT8. The mean real and imaginary parts are then taken over the kinematic training set and two ratios are calculated. That of the triply and doubly partitioned approximations and that of the approximations with a quartic and a triple partitioning. These two ratios are stored in a bipartite list, `PartReRat` in the case of the real part and `PartImRat` in the case of the imaginary part. For each variable, `j`, of each subsector, `k`, the partition set for the real part is then chosen according to:

$$\texttt{ChosenPartReSec[k]=}\begin{cases} \texttt{\{1,1,1,1,1,1\}, if (PartReRat[k][[2]]-} \\ \texttt{PartReRat[k][[1]])[[j]]<-0.1 ,} & (11.21) \\ \texttt{\{1,1,1\}, if -0.1<(PartReRat[k][[2]]-} \\ \texttt{PartReRat[k][[1]])[[j]]<0 ,} & (11.22) \\ \texttt{\{1\}, if (PartReRat[k][[2]] -} \\ \texttt{PartReRat[k][[1]])[[j]]>0 ,} & (11.23) \end{cases}$$

and likewise for the imaginary part,

$$\texttt{ChosenPartImSec[k]=}\begin{cases} \texttt{\{1,1,1,1,1,1\},if (PartImRat[k][[2]] -} \\ \texttt{PartImRat[k][[1]])[[j]]<-0.1 ,} & (11.24) \\ \texttt{\{1,1,1\}, if -0.1<(PartImRat[k][[2]]-} \\ \texttt{PartImRat[k][[1]])[[j]]<0 ,} & (11.25) \\ \texttt{\{1\}, if (PartImRat[k][[2]]-} \\ \texttt{PartImRat[k][[1]])[[j]]>0 ,} & (11.26) \end{cases}$$

With this, at the end of step OT9, all of the partitionings for each subsector have been chosen. They are assembled in the lists `ChosenPartReSec` and `ChosenPartImSec`.

11.6.10 OT10: Produce the Systematic Approximation

Along with the contour configurations in `CCFinal`, the partition sets in `ChosenPartReSec` and `ChosenPartImSec` are used in the TayInt calculation of general algebraic approximations which evaluate quickly to accurate and precise numerical approximations at any kinematic points.

11.6.11 Summary: OT1-10

All these quantitative steps and the criteria used within them are summarised in Tables 11.1 and 11.2. Thus, at the end of this section, all the steps in the final TayInt algorithm have been filled in and hence are all typeset in bold.

Table 11.1: Summary of the individual steps of the final TayInt algorithm, which have all now been presented.

U1: perform a sector decomposition on the finite integrals in the basis	
U2: locate all the thresholds of the Feynman integral	
Below threshold	Over threshold
BT1: $t_j \to y_j$	**OT1**: generate partition sets, full and partial contour configurations, kinematic training and cross-validation sets
BT2: Taylor expand the integrand and integrate	**OT2**: find partition:plain ratios for the full and partial contours
	OT3: Choose optimal full and partial contour by negative exclusion, backup with positive selection
	OT4: Perform kinematic cross validation
	OT5: Assess the optimal full and partial contours and choose between them
	OT6: Generate the uniform partition ratios on the chosen contours
	OT7: Assess the accuracy of the chosen contours to decide if a high-uniform partitioning can be used for all subsectors
	OT8: If not, use the accuracy and improvement assessment to decide if a low- or high-uniform partitioning is appropriate
	OT9: If not, analyse the subsectors on a per-variable basis and find the optimal varied partitioning
	OT10: Taylor expand and integrate

Table 11.2: Summary of the selection criteria used in the quantitative analysis of the over-threshold part of the final TayInt algorithm. In the contour selection part of the algorithm (OT3-5), the grey rows denote the layer L2, in which the partial contours are analysed. In the partition selection part of the final TayInt algorithm (OT7-9), the grey rows denote the layer L4, in which the varied partition sets are analysed.

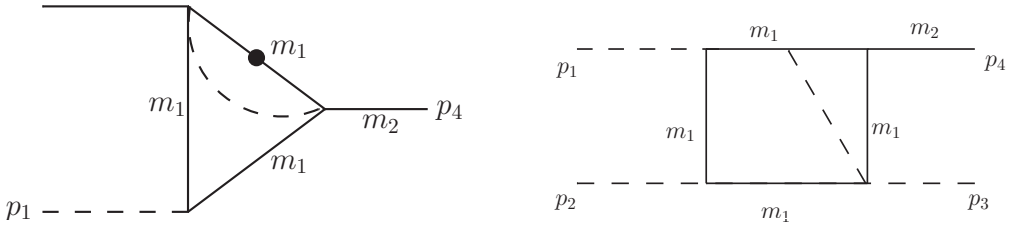
Step	Selection Criteria
L1 & L2: Contour Selection	
OT3	$0.5 <$ `TrainRatSensFracSec[k]` < 1.5
OT3	$0.5 <$ `PartialTrainRatSensFracSec[k]` < 1.5
OT4	$[($`ContCrossVal1[[k]]` $< 0.5) \cap ($`ContCrossVal2[[k]]` $< 0.25) \cap$ ($`ContCrossVal3[[k]]` $< 0.5)] \cap [$`TrainedContourSec[k]`$] \neq \emptyset$
OT4	$[($`PartialContCrossVal1[[k]]` $< 0.5) \cap ($`PartialContCrossVal2[[k]]` $< 0.25) \cap$ ($`PartialContCrossVal3[[k]]` $< 0.5)] \cap [$`PartialTrainedContourSec[k]`$] \neq \emptyset$
OT5	`FuParComp[[k]] = If[FuParProx[[k]]>0.15, TrainFuParComp[[k]], CrossFuParComp[[k]]]`
OT5	$0.5 <$ `FuParComp[[k]]` < 1.25
OT5	$0.8 <$ `FuParComp[[k]]` < 1.2
OT5	`Max[Length[(0.5 < FuParComp[[k]]<0.7)]-Length[( FuParComp[[k]]<0.5) ∪ ( FuParComp[[k]]>1.25)]]`
L3 & L4: Partition Selection	
OT7	$0.95 <$ `Mean[Hom3Hom2AbsRat[k]]` < 1.05
OT8	$[(0.95 <$`Hom2PlainAbsRat[[k]]` $< 1.05) \cap (0.95 <$`Hom3PlainAbsRat[[k]]` $< 1.05) \cap$ ($0.95 <$`Hom3PlainReRat[[k]]` $< 1.05) \cap (0.95 <$`Hom3PlainImRat[[k]]` $< 1.05) \cap]$
OT8	$0.4 <$ `Mean[Hom3PlainAbsRat]` < 0.6 OR $0.4 <$ `Mean[Hom2PlainAbsRat]` < 0.6
OT9	`Mean[Hom3PlainAbsRat]` > 1.1 && `Mean[Hom2PlainAbsRat]` > 1.1

OT9	`(PartReRat[k][[2]] - PartReRat[k][[1]])[[j]] < -0.1`
	`(PartReRat[k][[2]] - PartReRat[k][[1]])[[j]] < 0`
	`(PartReRat[k][[2]] - PartReRat[k][[1]])[[j]] > 0`

OT9	`(PartImRat[k][[2]] - PartImRat[k][[1]])[[j]] < -0.1`
	`(PartImRat[k][[2]] - PartImRat[k][[1]])[[j]] < 0`
	`(PartImRat[k][[2]] - PartImRat[k][[1]])[[j]] > 0`

11.7 Calculation Code

Now that the quantitative backbone of the final TayInt algorithm has been described, the complete algorithm must be demonstrated from start to finish. But first the method of calculating the Feynman integral has to be understood because the TayInt algorithm in its final form follows the principle of mimicry. In the remainder of this chapter, the results of applying the TayInt code to the two-loop Feynman integrals I10 and I59 (elliptic) will be frequently discussed, so these integrals are depicted in Fig. 11.2.



(a) I10 (b) I59

Figure 11.2: The two-loop triangle I10 graph for which analytical results are available [92] and the elliptic I59, (b), thus far unknown analytically. Dashed lines indicate massless particles, solid internal lines massive particles, and dots denote squared propagators. Solid external lines denote massive and else off-shell particles.

11.7.1 Algebraic

Many Feynman integrals, such as I59, are immensely complicated and it is not always possible to fully and generally integrate them analytically, although excellent progress is being made, see for example [32]. However, a polynomial representation, obtained by means of a Taylor expansion in the variables of integration, can always be integrated, regardless of the mathematical complexity of the original integrand. Moreover, the integrand still retains its algebraic dependence on all of its parameters after being Taylor expanded. These parameters are manipulated using the contour configurations and partition sets determined by the final TayInt algorithm, to calculate systematic approximations for a generic Feynman integral. In order to do this, three steps are necessary, which use three Mathematica functions defined in Appendix B.1. The first, performed by `COMPDiscTaySeriesIntegrator`, is to produce an integrated Taylor expansion for each subsector of the Feynman integral. The function `COMPDiscTaySeriesIntegrator` Taylor expands and integrates the subsector of the Feynman integral, with *algebraic integration boundaries and expansion points* and outputs a `.txt` file containing each term in the integrated Taylor expansion. There are three features of the `COMPDiscTaySeriesIntegrator` module that are crucial to the efficiency of the TayInt code:

- **The use of `Map`:** all iterated operations are performed using the `Map` operator applied to lists (known as vectorisation), which is much faster than using loops or the `Table` command in Mathmatica.

- **Minimising the argument of `Integrate`.** The most computationally intensive tasks are performed as little as is mathematically possible. The integration step is the most time consuming part of the `COMPDiscTaySeriesIntegrator` module. Therefore, only the parts of the integrand that *must* be integrated are passed to the `Integrate` function and they are simplified as much as possible. As the compexity of the subsectors of the Feynman integral is transferred from the Feynman parameters to the coefficients of the Taylor expansion, excluding these coefficients

reduces the run time of this part of the code by a factor of 3160, or a reduction of 99.968%.

- **Sacrificing efficiency now for greater gains later.** It is faster to output the algebraic Taylor integrated approximation all at once rather than splitting it up, because this minimises the number of times files need to be written. However, when the expansion points and limits of integration are inserted in the next step to generate the algebraic approximations in each partition of the integrand, it is much faster to proceed on a piece-by-piece basis, minimising the sizes of the files that need to be processed by the insertion operation. Hence, looking ahead, each term in the integrated Taylor expansion is stored in a separate file.

With the approximations stored in this form, the function `COMPDiscTaySliceOutput`, then takes these order-by-order files as input and replaces the algebraic integration boundaries and expansion points by the boundaries and midpoints of each partition of the integrand that has been determined by the final TayInt algorithm. The direction of the the relevant limits of integration are determined by the contour configuration (OT2-5) and their size is determined by the partition set prescribed by the final TayInt algorithm (OT6-9). The midpoints of these algorithmically determined integration regions are used as the expansion points. Each of these slices are subsequently simplified and written to `.txt` files, in which form they are stored as output. This generates an approximation for the Feynman integral that is algebraic in the kinematic scales and is primed to produce accurate and precise numerical approximations at any kinematic point.

> These files are the main achievement of TayInt, as they are algebraic in the kinematic scales so can be quickly evaluated and summed together to give accurate and precise algebraic approximations for the Feynman integral valid at any kinematic point.

Adhering to the mimicry principle (P1), the two functions `COMPDiscTaySeriesIntegrator` and `COMPDiscTaySliceOutput` are also used by the final TayInt algorithm to produce the approximations needed to construct the ratios in Eqs. (11.7)-(11.18). These are used for the quantitative analysis of the subsectors of Feynman integrals, as described in the preceding Section 11.6.

11.7.2 Numerical

The algebraic approximations generated as part of the TayInt calculation can then be passed to the function `COMPKine` which inserts the specified kinematic points to generate precise and accurate numerical approximations for the starting Feynman integral. The details of the computation of the algebraic and numerical approximations is demonstrated in detail in Appendix B.1.

11.8 Application of the Final TayInt Algorithm

The structure of the final TayInt algorithm has been illustrated in Fig. 11.1 and its quantitative tools specified and summarised in Table 11.2. This section illustrates the final TayInt algorithm step-by-step by its application to the elliptic I59 Feynman integral at order ϵ^1 and to the two-loop I10 integral.

11.8.1 Illustrating the Objectives

Slices of the integrands of I10 and I59 are plotted in the Figs. 11.3, 11.4, 11.5,11.6, 11.7 below. This illustrates why the changes C1-3 described at the beginning of this chapter are needed to bring elliptic integrals within the capability of TayInt,

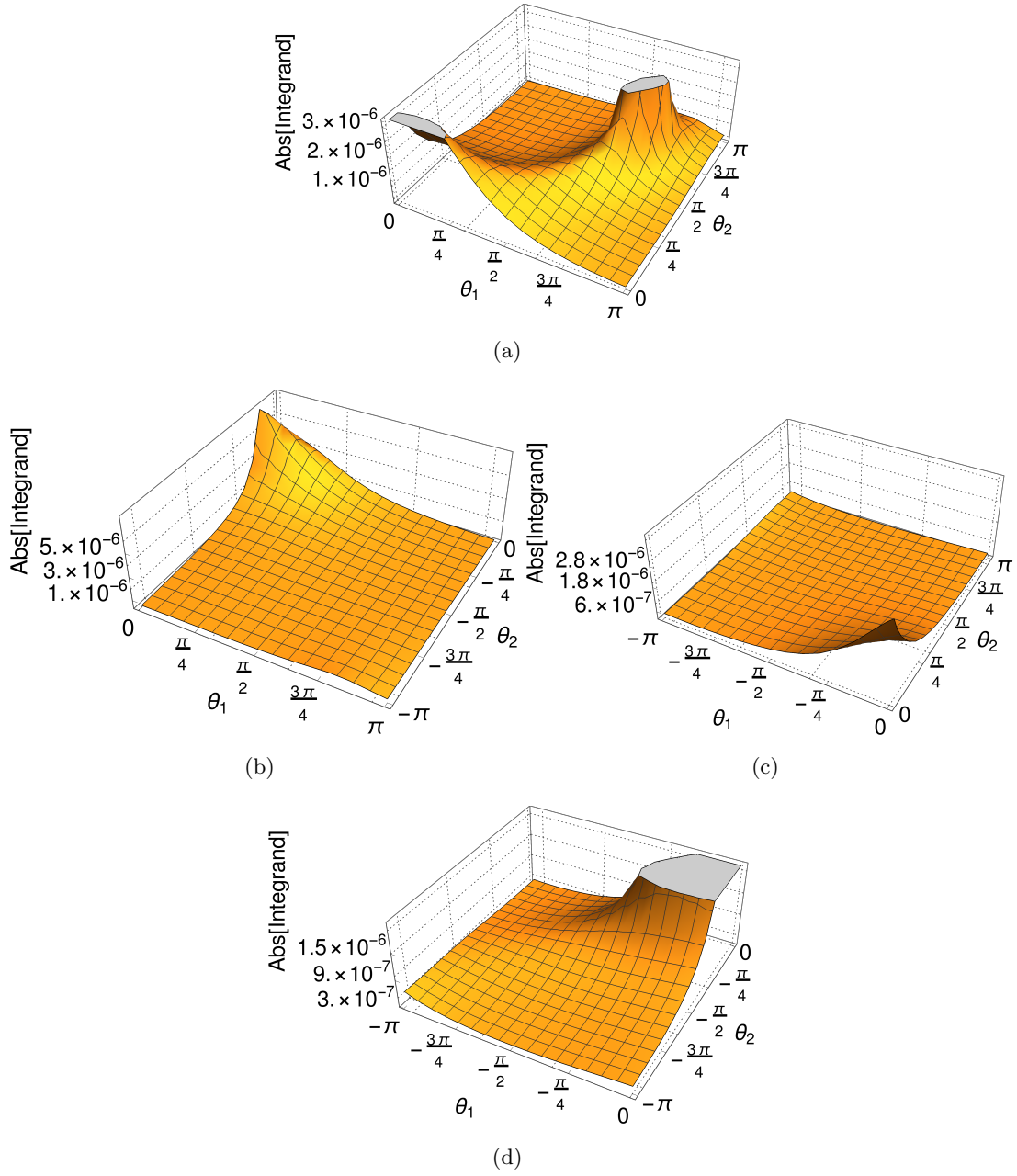


Figure 11.3: A slice of the absolute value of the $I10_4$ integrand at ϵ^1 is shown after a complex mapping in four of the eight possible contour configurations. The kinematic scales are set to $u = 10.4m_1^2$, $m_2 = \frac{m_1}{\sqrt{2}}$ and $m_1 = 173$ GeV. The smooth nature of the surfaces in Figure (b), (c) and (d), all of which are suitable for use in the TayInt calculation indicates the comparative ease in algebraically approximating the subsectors of non-elliptic integrals with TayInt, hence I10 could be calculated using the conceptual TayInt algorithm.

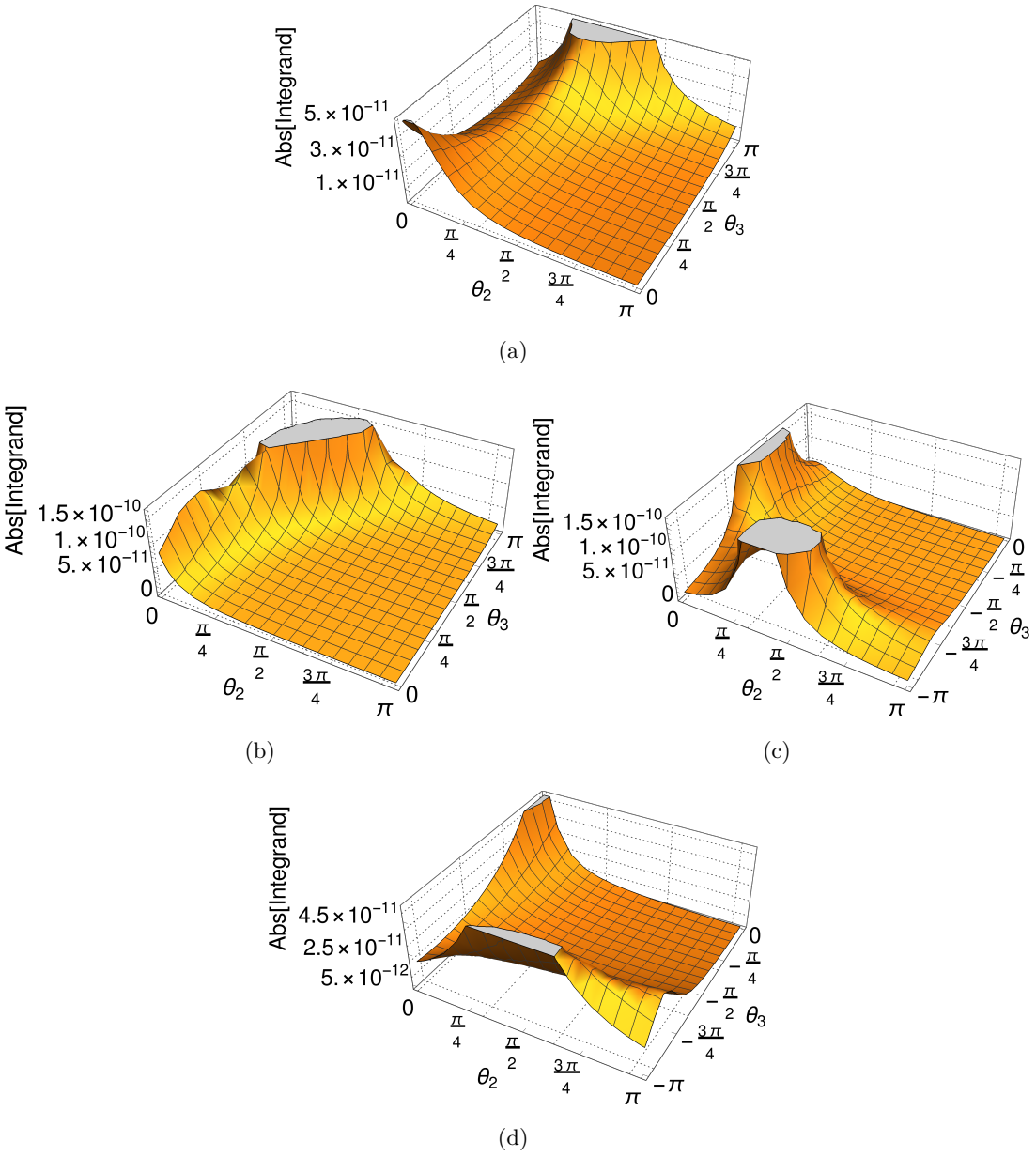


Figure 11.4: A slice of the absolute value of the $I59_4$ integrand at ϵ^1 is shown after a complex mapping in four of the 32 possible contour configurations. The kinematic scales are set to $s = 10.4m_1^2$,$u = -2m_1^2$, $m_2 = \frac{m_1}{\sqrt{2}}$ and $m_1 = 173$ GeV. The fluctuating nature of the surfaces, means that only one of which (Figure (a)) is suitable for use in the TayInt calculation. Comparing this situation to that depicted in Fig. 11.3 for $I10_4$ indicates the comparative difficulty in algebraically approximating the subsectors of elliptic integrals with TayInt, hence the need for change C1.

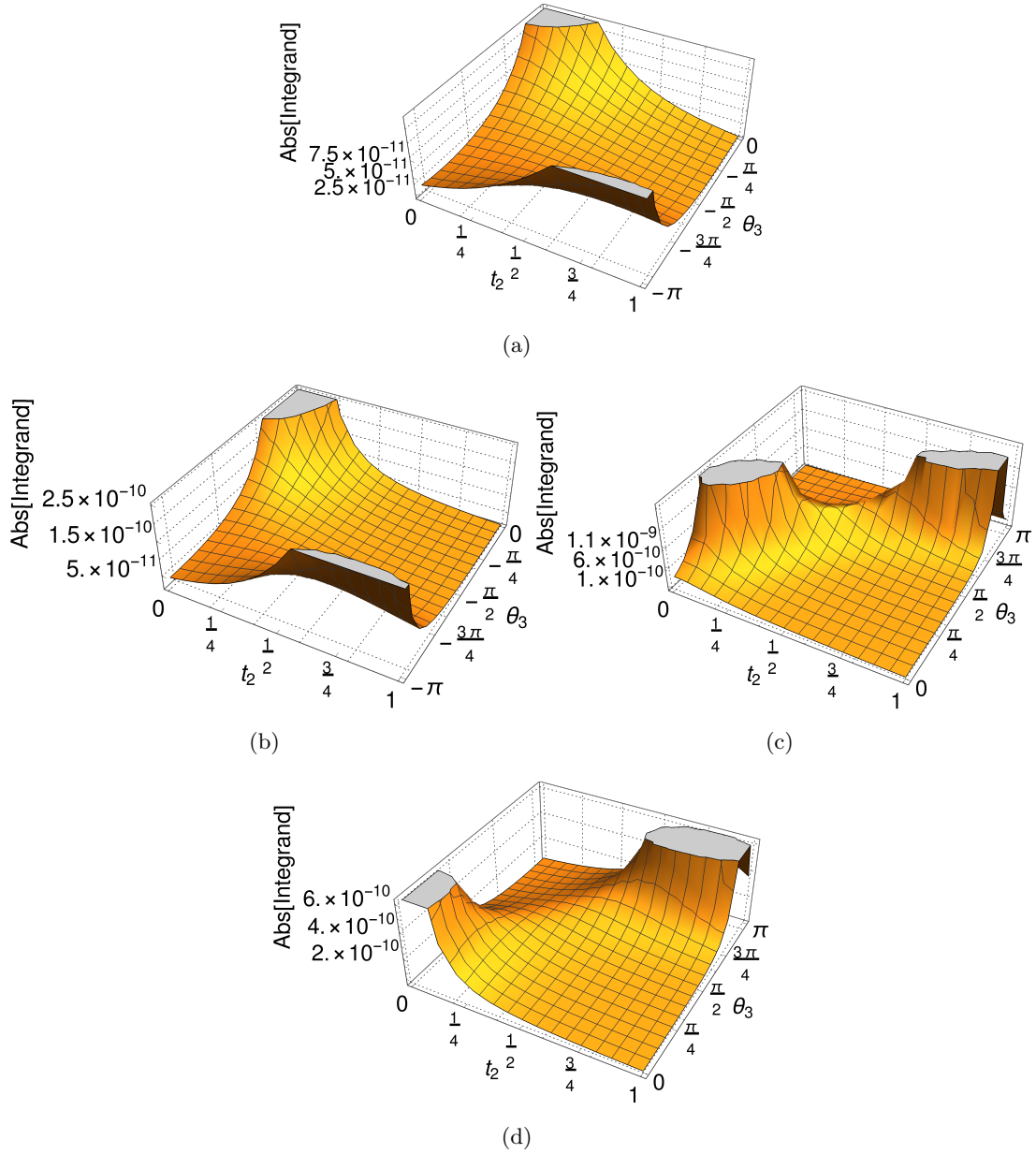
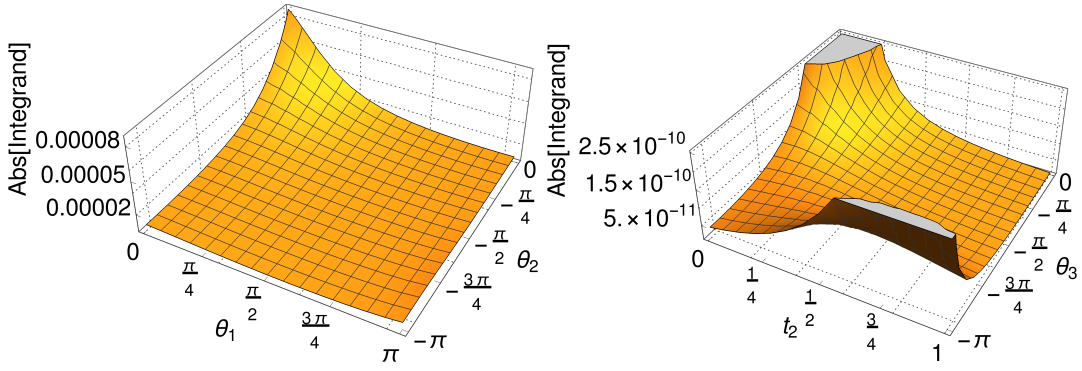
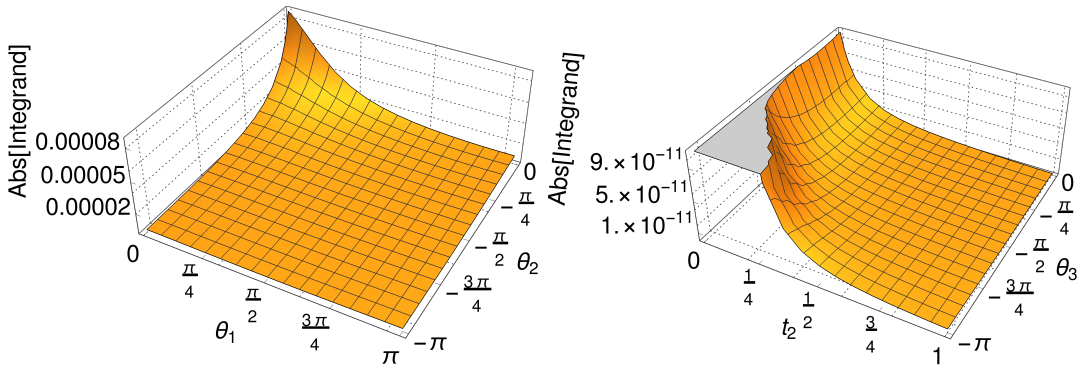


Figure 11.5: A slice of the $I59_4$ integrand at ϵ^1 is shown after a partial complex mapping in four of the eight possible contour configurations for that particular mapping. The kinematic scales are set to $s = 10.4m_1^2$,$u = -2m_1^2$, $m_2 = \frac{m_1}{\sqrt{2}}$ and $m_1 = 173$ GeV. The smoother nature of the surfaces depicted in (a) and (b) compared to those shown in Fig. 11.4 indicates the importance of including the partial contours in implementing change C1.



(a) (b)

Figure 11.6: Slices of the absolute values of the (a) $\mathrm{I10}_4$ and (b) $\mathrm{I59}_4$ integrand at ϵ^1 are shown using the contour configuration chosen by the final TayInt algorithm. The kinematic scales are set to $u = 10.4m_1^2$, $m_2 = \frac{m_1}{\sqrt{2}}$, $m_1 = 173$ GeV and $s = 10.4m_1^2$,$u = -2m_1^2$, $m_2 = \frac{m_1}{\sqrt{2}}$ and $m_1 = 173$ GeV respectively. The smoother nature of the surface depicted in Figure (a) compared to that shown in Figure (b) indicates the importance of combining varied and uniform partition sets in implementing change C2.



(a) (b)

Figure 11.7: Slices of the absolute value of the (a) $\mathrm{I10}_4$ and (b) $\mathrm{I59}_4$ integrand at ϵ^1 are shown using the contour configuration chosen by the final TayInt algorithm. The kinematic scales are set to $u = 20m_1^2$, $m_2 = \frac{m_1}{\sqrt{2}}$, $m_1 = 173$ GeV and $s = 20m_1^2$,$u = -2m_1^2$, $m_2 = \frac{m_1}{\sqrt{2}}$ and $m_1 = 173$ GeV respectively. The greater change relative to Fig. 11.6 of the surface depicted in Figure (b) compared to that shown in Figure (a) indicates the importance of using training and cross-validation sets of kinematic points in implementing change C3.

Having illustrated the need for the changes C1-3, the application of each step of

the over-threshold part of the final TayInt algorithm that implements them is now demonstrated using the two aforementioned Feynman integrals, I10 and I59.

11.8.2 OT1: Generate the Partition Sets, Contour Configurations and Kinematic Training and Cross-Validation Sets

In the first step, OT1, all the information needed to work with the Feynman integral is assembled. This information can be broken down into several categories, as follows.

Relevant User Input

1. The relevant information from the user, for example the kinematic scales of the integral and the list of kinematic points at which the user wants results, are imported. In the case of I59, these are

Example 11.8.1

```
kineinv={q12s,q23s,q123s}                                    (11.27)
ResKine={{119716.,-59858.,14964.5,29929.},...,
        .....,{239432.,-59858.,14964.5,29929.}},             (11.28)
```

   where the list of kinematic invariants corresponds to s, u, m_h^2 and the `ResKine` list contains an example of the values that could be requested by the user for each of these invariants, grouped into sub-lists.

Generate the Kinematic Training and Cross-Validation Sets

2. This step is crucial to the success of the TayInt method, as it ensures that the algebraic approximations it generates can be evaluated quickly to yield precise numerical approximations at *any* kinematic point. In order to find the appropriate contours that guarantee this and achieve objective O3 of TayInt, the subsectors need to be analysed using a training set of kinematic points. The optimal contours must then be validated using three further cross-validation sets, as described in section 11.6.

Import the Subsectors and Generate the List of Feynman Parameters

3. The subsectors are generated in step U1 of the final TayInt algorithm using the code from SecDec-3.0.9. They are then imported and their Feynman parameters extracted and stored in a list. To give an example of the scope of the problem, the

first subsector of the integral I59 at order ϵ^1 reads:

$$\begin{aligned}
&\mathrm{I59}^1_{01} = (\log[t_4] - 3\log[1 + t_0 + t_1 + t_2 + t_3 + (1 + t_0)\cdot(t_1 + t_2 + t_3)t_4] \\
&2\log[-s\cdot t_3\cdot(1 + t_1 + t_1t_4) - t_0(u\cdot t_2 + s\cdot t_1t_3t_4) \\
&-u\cdot t_0\cdot(1 + t_1 + (t_1 + t_2 + t_3)\cdot t_4) \\
&+m^2\cdot(1 + t_0 + t_1 + t_2 + t_3)\cdot(1 + t_0 + t_1 + t_2 + t_3 + (1 + t_0)\cdot(t_1 + t_2 + t_3)\cdot t4)]\Big) \\
&(u\cdot t_0t_2 + s\cdot t_3 + s\cdot t_1t_3 + s\cdot(1 + t_0)t_1t_3t_4 \\
&+m_h^2\cdot t_0(1 + t_1 + (t_1 + t_2 + t_3)\cdot t_4) \\
&-m^2\cdot(1 + t_0 + t_1 + t_2 + t_3)(1 + t_0 + t_1 + t_2 + t_3 + (1 + t_0)\cdot(t_1 + t_2 + t_3)\cdot t_4)\Big)^{-2},
\end{aligned} \tag{11.29}$$

from which the list of Feynman parameters is extracted as `FeynList`$= \{t_0, t_1, t_2, t_3, t_4\}$.

Transform to the Space of Complex Feynman Parameters and Generate the List of Possible Full and Partial Contours

4. As described in Section 11.6, the subsectors of the Feynman integral are transformed to the space of complex functions as in such a space there are two ways to perform a univariate integral. The effect of this is to transform from one possible integration contour, to 2^j, or 32 in the case of I59, for which $j = 5$. All these potential contours are stored in the list `CCNIELcalc` (defined in Eq. (11.1)). The position of the optimal contour in this list will be determined in the first layer (L1) of the the final TayInt algorithm, steps OT2-OT5.

Generate the List of Partial Complex Contours

5. In accordance with the principle of maximising information (P2), the nested list of the subsectors of the Feynman integral after all possible partial complex mappings (see Eqs. (11.2)-(11.3)) is also generated, along with the corresponding nested list of partial contour configurations. This list is termed `CCNIELpartialcalc` and the position of the optimal contour in this list will be determined in the second layer (L2) of the the final TayInt algorithm, steps OT2-OT4. In the case of the highly complicated I59, this list contains 40 possible contour configurations, considerably increasing the possibility of finding a contour which is well suited to a Taylor expansion. This extra information is especially important when it is very difficult to find a representation of the subsectors of a Feynman integral (such as I59) that is suitable for a Taylor expansion. It constitutes part of the implementation of change C1 to tackle more complicated Feynman integrals.

Generate the List of Test Partitions

6. Section 11.6 noted that the effects of performing a partitioning must be recorded to assess the suitability of a contour for producing precise, accurate approxima-

tions with TayInt. Therefore, the final part of step OT1 is to generate the lists of partitions used to test the subsectors on the different contours in `CCNIELcalc` and `CCNIELpartialcalc`. This is in adherence with the mimicry principle of TayInt (P1). In the case of I10 (which has three integration variables after sector decomposition and so is easier to visualise than I59), these testing lists read:

Example 11.8.2

```
NoPart={{1},{1},{1},{1},{1}}                                    (11.30)
TestPart=                                                       (11.31)
{{{1,1,1},{1},{1},{1},{1}},
{{{1},{1,1,1},{1},{1},{1}},
{{{1},{1},{1,1,1},{1},{1}},
{{{1},{1},{1},{1,1,1},{1}},
{{{1,1,1},{1},{1},{1},{1,1,1}}} .
```

At the end of step OT1, the complex-mapped subsectors, their lists of Feynman parameters and kinematic scales and the lists needed to test and find a TayInt-suitable representation of them are all ready for use in the following steps.

11.8.3 L1 (OT2): Generate the Partition:Plain Ratios on the Full Contours

The goal of this section (OT2-4) of the over-threshold algorithm is to find the position of the contours in `CCNIELcalc` and `CCNIELpartialcalc` which are most suitable for use in the TayInt calculation. *Suitable* means a contour which is an accurate representation of the subsector which yields improved precision and accuracy when the integrand is partitioned, in accordance with objectives O2 and O3 of TayInt. As outlined in Fig. 11.1 and in adherence with the principle of maximising information (P2), the steps OT2-OT4 are performed using both the full and partial contour configurations contained in `CCNIELcalc` and `CCNIELpartialcalc` respectively. In this way, the final TayInt algorithm can be split into two layers, L1 for the full contours and L2 for the partial contours.

`CCNIELcalc` is a list of contour configurations for one complex transformation, that of mapping all the Feynman parameters to complex parameters. `CCNIELpartialcalc` is a nested list of contour configurations for each possible partial complex transformation. Thus, the position of the optimal contour is given by a number in the first case and by a bipartite list in the second. However, as the mathematial procedure is equivalent, steps OT2-4 will only be demonstrated for the first layer (L1). The first part of step OT2 is to generate the algebraic integrated Taylor expansion using the parameters specified by the user in `COMPDiscTaySeriesIntegrator`, a function described in detail in Appendix B.1:

```
COMPDiscTaySeriesIntegrator[CAdomsec[[#]],varlist,varlist,0,
4,(ToString[name]<>"OTEPS"<>ToString[epsord]),#]&/@
Range[1,Length[SecList]] .
```

Integrated Taylor expansions with no partitions, given by `COMPDiscTaySliceOutput` with `ralist=NoPart` and with each integration variable partitioned, given by `COMPDiscTaySliceOutput` with `ralist=TestPart`, are produced from these algebraic approximations. Numerical approximations, `TrainGenRatSensKineSec[l][k]][[j]][[i]]`, `Cross1GenRatSensKineSec[l][k]][[j]][[i]]`, `Cross2GenRatSensKineSec[l][k]][[j]][[i]]`, `Cross3GenRatSensKineSec[l][k]][[j]][[i]]` are then generated in turn. Here, `l` runs over the subsectors, `k` over each possible contour configuration from the list `CCNIELcalc`, Eq. (11.1), `j` over the kinematic points in the relevant training or cross-validation set and `i` over each partition set in `TestPart`, Eq. (11.31). The former are the approximations generated using the training set, with which the optimal contours will be selected. The latter are generated using the smaller cross-validation sets, which will be used to check that the optimal contours generalise to different kinematic points. Each such set of approximations is then divided by the corresponding approximations obtained similarly but using `NoRat`. This was demonstrated in Equation 11.7, where the Mean Absolute value is taken over the range of kinematic points. For I59 subsector 1 at order ϵ^1, the training approximations read:

Example 11.8.3

$$
\begin{aligned}
&\texttt{Table[TrainRatSensFracSec[1]=}\\
&\{\{0.238,0.380,0.170,0.115,262.587\},\\
&\{0.377,29.947,0.1545,0.119,169.319\},\\
&\{\ \vdots\ ,\ \vdots\ ,\ \vdots\ ,\ \vdots\ ,\ \vdots\ \},\\
&\{0.915,1.012,0.848,1.108,0.838\},\\
&\{1.229,0.773,1.361,1.026,0.8455\}\\
&\{\ \vdots\ ,\ \vdots\ ,\ \vdots\ ,\ \vdots\ ,\ \vdots\ \}\}\ .
\end{aligned}
\tag{11.32}
$$

11.8.4 L1 (OT3): Choose the Full Contours via Negative Exclusion

The 32 inner lists in Eq. (11.32) contain the partition:plain ratios for each variable of I59 subsector 1 at order ϵ^1. These are found using each contour configuration in `CCNIELcalc`, leading to the outer list. The numbers correspond to inserting the points in the kinematic training set and computing the mean absolute value. In step OT3, The principle of negative exclusion (P3) is applied to select the optimal contour. This proceeds as follows.

Finding the Lists of Contours with No Unsuitable Variables: Negative Exclusion

Instead of examining the suitability of each integration variable individually, negative exclusion looks at the suitability of the full set of variables as a whole. This is best illustrated by the I10 integral at order ϵ^0. In this case, the maximum number of individual integration variables well suited (partition:plain ratio in the range $[0.25, 1]$) for a partitioning is one. In this way, the fourth contour configuration would be selected, as this has the lowest mean partition:plain ratio of those contours with one well suited variable. However, the fourth entry of `TrainRatSensFracSec[1]`, corresponding to the fourth contour configuration, reads `{11.309,2.133,0.765}`. Viewed entirely in terms of the best individually suited integration variable, with ratio 0.765, this is the suitable choice. But, viewed in light of all the integration variables it is clearly unsuitable. The third entry of `TrainRatSensFracSec[1]`, corresponding to the third contour configuration, reads `{1.051, 1.195, 1.013}`. As an overall set of ratios this is much more suitable for a Taylor expansion than that corresponding to the fourth contour configuration. Therefore, the main procedure for choosing the optimal contour seeks the contour which does not have any integration variables unsuitable for a partitioning (with ratios less than 0.5 or greater than 1.5), termed negative exclusion. First, the TayInt algorithm finds a list of the positions of the integration variables which have partition:plain ratios lower than 0.5, for each contour of each subsector, labelled by `k`:

```
Table[TrainSensBadLowVarPosSec[k]=
Table[Flatten[
Position[TrainRatSensFracSec[k][[i]],_?(#<0.5&)]],
{i,1,Length[TrainRatSensFracSec[k]]}],
{k,1,Length[SecList]}] ,                                    (11.33)
```

or greater than 1.5;

```
Table[TrainSensBadHighVarPosSec[k]=
Table[Flatten[
Position[TrainRatSensFracSec[k][[i]],_?(#>1.5&)]],
{i,1,Length[TrainRatSensFracSec[k]]}],
{k,1,Length[SecList]}] .                                    (11.34)
```

It subsequently finds the length of these two lists:

```
Table[TrainLenBadLowVarPosSec[k]=
Length[TrainSensBadLowVarPosSec[k][[#]]]&/@
Range[1,Length[TrainSensBadLowVarPosSec[k]]],
{k,1,Length[SecList]}] ,                                    (11.35)
```

```
Table[TrainLenBadHighVarPosSec[k]=
Length[TrainSensBadHighVarPosSec[k][[#]]]&/@
Range[1,Length[TrainSensBadHighVarPosSec[k]]],
{k,1,Length[SecList]}] .                                    (11.36)
```

The final TayInt algorithm then locates the contours for which these lists have zero length:

```
Table[TrainNullLenBadLowVarSensPosSec[k]=
Flatten[Position[TrainLenGoodVarPosSec[k],
0,{k,1,Length[SecList]}],                                   (11.37)
```

```
Table[TrainNullLenBadHighVarSensPosSec[k]=
Flatten[Position[TrainLenBadHighVarPosSec[k],
0,{k,1,Length[SecList]}],                                   (11.38)
```

then finds the position of their intersection;

```
Table[TrainNullLenBadVarSensPosSec[k]=
Intersection[TrainNullLenBadHighVarSensPosSec[k],
TrainNullLenBadLowVarSensPosSec[k]],{k,1,Length[SecList]}] .  (11.39)
```

This returns `TrainNullLenBadVarSensPosSec[k]`, the list of contours for each subsector `k` which are free of any integration variables with partition:plain ratios in the range $[0.5, 1.5]$. This range is used to select accurate contours, numerically represented by ratios that are insensitive to the partitioning and hover around 1, producing a similar approximation irrespective of the partition depth.

Choosing the Optimal Contours in Negative Exclusion

The next step is to compute the mean over the integration variables for each of the potential contours selected by negative exclusion and select that which minimises it.

This is done using the following code:

```
Table[TrainMeanRatBadVarSensFracSec[k]=
Table[Mean[TrainRatSensFracSec[k]
[[TrainNullLenBadVarSensPosSec[k][[i]]]]],
{i,1,Length[TrainNullLenGoodVarSensPosSec[k]]}],
{k,1,Length[SecList]}],                                    (11.40)
Table[TrainMinMeanRatBadVarSensFracSec[k]=
Flatten[Position[TrainMeanRatBadVarSensFracSec[k],
Min[TrainMeanRatBadVarSensFracSec[k]]]][[1]],
{k,1,Length[SecList]}],                                    (11.41)
Table[TrainChosenBadVarRatContourSec[k]=
TrainNullLenBadVarSensPosSec[k]
[[TrainMinMeanRatBadVarSensFracSec[k]]],
{k,1,Length[SecList]}],                                    (11.42)
```

returning `TrainChosenBadVarRatContourSec[k]`, a list of numbers denoting the position in `CCNIELcalc` of the contour with the lowest mean partition:plain ratio of those with no unsuitable variables for the TayInt calculation. The mean is minimised because this quantifies the extent to which the approximation is improving by increasing the number of partitions, which is the next criterion for suitable contours after accuracy (see objectives O1 and O2). The partition:plain ratios being in the range $[0.5, 1.5]$ suggests that the contour is an accurate representation of the subsector, the contour with the lowest mean ratio from amongst them gains the most precision when partitioned. In this way, a contour configuration that will yield accurate numerical approximations whose precision can be increased is found, achieving objectives O2 and O3 of TayInt.

If no contour can be found using negative exclusion for a particular subsector, then the contour which has the most partition:plain ratios which are improving quickly is selected, termed positive selection. The code is very similar to that given in Eqs. (11.33)-(11.42), except that the desired range for the partition:plain ratios is $[0.25, 1]$. The lower bound is set to 0.25 because partition:plain ratios less than 0.25 indicate that on this contour the Taylor expansion is converging to the wrong value. This is because a very small ratio shows that the plain TayInt approximation is very large and so the choice of contour is not an accurate representation of the integrand.

In the case of I10 at order ϵ^0, the positive selection (quickly improving) and negative exclusion (accurate) contour choices, given by
`TrainChosenGoodVarRatContourSec[k]` and `TrainChosenBadVarRatContourSec[k]`, read:

Example 11.8.4

```
Table[TrainChosenGoodVarRatContourSec[k]=
TrainMaxLenGoodVarSensPosSec[k]
[[TrainMinMeanRatGoodVarSensFracSec[k]]],
{k,1,Length[SecList]}],                                    (11.43)
{4,7,2,1,8,2,1,4}
Table[TrainChosenBadVarRatContourSec[k]=
TrainNullLenBadVarSensPosSec[k]
[[TrainMinMeanRatBadVarSensFracSec[k]]],
{k,1,Length[SecList]}]                                     (11.44)
{3,7,4,3,8,2,1,2},
```

with the difference in selection for subsector 1 illustrating the importance of using negative exclusion (P3).

Choosing the Optimal Contour from the Positive Selection and Negative Exclusion Sets

In accordance with this principle of the TayInt algorithm (P3), the optimal full and partial contours for each subsector are those optimised with negative exclusion: the accurate contour with the most potential for precision gain. If no contour can be found using negative exclusion, then that optimised with positive selection (fastest improvement) is taken instead:

$$\texttt{TrainedContourSec[k]=}\begin{cases}\texttt{TrainChosenGoodVarRatContourSec[k],}\\ \texttt{if Length[TrainNullLenBadVarSensPosSec[k]}\\ \texttt{== 0,}\\ \texttt{TrainChosenBadVarRatContourSec[k],}\\ \texttt{otherwise ,}\end{cases} \tag{11.45}$$

where the index `k` runs over the subsectors. In L1 (full contours) this is a list of numbers, in L2 (partial contours) it is a list of bipartite lists, termed `PartialTrainedContourSec[k]`, as the transformation and the contour configuration within it must be specified. These lists contain those contours which have been tried and tested against the demands of accuracy and precision improvement and come through successfully, enforcing the demands of objectives O1-2. They must next be tested against the demand of O3, kinematic generalisation.

11.8.5 L1 (OT4): Perform Kinematic Cross Validation

In both the full and partial cases, the optimal contours have been selected in step OT3 using the training set of kinematic points. To achieve objective O3 and ensure that these contours can also generate accurate and precise approximations at a different set of kinematic points, kinematic cross validation must be performed. This is done by computing the relative mean absolute difference between the training partition:plain ratios and each of the cross-validation partition:plain ratios, `TrainRatSensFracSec[k]` and `Cross1RatSensFracSec[k]` etc. The contours which have such a difference within 0.5, 0.25, 0.5, in the three respective pairings, are deemed valid. The difference in the acceptance threshold is because in the first and third cross-validation sets the approximations are naturally worse, since these sets contain kinematic points close to the threshold or very far above it, so a more generous margin is needed. This comparison is carried out using the quantitative tool introduced in Eq. (11.9), encoded as:

```
ContCrossVal1=Mean[Abs[Transpose[
Cross1RatSensFracSec[#]]-Transpose[TrainRatSensFracSec[#]]]]
          Abs[Transpose[TrainRatSensFracSec[#]]]
&/@ Range[1,Length[SecList]] ,                                    (11.46)
Table[LowCrossVal1[k]=
Flatten[Position[ContCross1Val[[k]],_?(#<0.5&)]],
{k,1,Length[SecList]}] ,                                          (11.47)
Table[LowCrossVal2[k]=
Flatten[Position[ContCross2Val[[k]],_?(#<0.25&)]],
{k,1,Length[SecList]}],                                           (11.48)
Table[LowCrossVal3[k]=
Flatten[Position[ContCross3Val[[k]],_?(#<0.5&)]],
{k,1,Length[SecList]}] ,                                          (11.49)
Table[LowCrossVal[k]=
Intersection[LowCrossVal1[k], LowCrossVal2[k], LowCrossVal3[k]],
{k,1,Length[SecList]}] ,                                          (11.50)
Table[IntTrainCrossContSec[k]=
Intersection[Flatten[TrainedContourSec[k]], Flatten[LowCrossVal[k]]],
{k,1,Length[SecList]}] .                                          (11.51)
```

If there is an intersection between these triply cross-validated contours, given in `LowCrossVal[k]` and the contour selected using the training set, `TrainedRatContourSec-[k]`, that contour is deemed valid, stored in the list `IntTrainCrossContSec[k]` and used as the optimal contour for the subsector `k` within L1. The corresponding list in L2 is termed `PartialIntTrainCrossContSec[k]`, for each subsector `k`. In either layer, if the intersection is empty, then the mutual set of contours with a relative cross-validation

difference of less than the required threshold in `LowCrossVal[k]`, is generated with the code:

```
TrainNumLowCrossVal=If[Length[LowCrossVal[#]]!=0,
Table[TrainRatSensFracSec[#][[LowCrossVal[#][[i]]]],
{i,1,Length[LowCrossVal[#]]}]]
&/@Range[1,Length[SecList]],                                    (11.52)
```

then the same procedure of negative exclusion and positive selection (OT2-3) is performed on this subset of contours only.

The contour selected to represent the `k`th subsector using negative exclusion and positive selection from the set with a low cross-validation error is called `TrainCrossValContourSec[k]`. It is used as the final choice of contour, `ChosenContourSec[k]`, in the event that there is an empty intersection between the contour selected using the training set and the contours with a low cross-validation error. This entire multi-layered decision process may be summarised as follows:

$$\texttt{ChosenContourSec[k]} = \begin{cases} \texttt{TrainedContourSec[k]}, \\ \texttt{if Length[IntTrainCrossContSec[k]]} \neq 0 \\ \texttt{TrainCrossValContourSec[k]}, \\ \texttt{otherwise} \end{cases} \tag{11.53}$$

In the case of I10 at order ϵ^0 a full contour can always be found by negative exclusion and so this list reads

Example 11.8.5

```
ChosenContourSec[k]={3,7,4,3,8,2,1,2},                          (11.54)
```

This ensures that the accuracy and precision of the optimal contour is not tied to a particular region and generalises to different kinematic points, in accordance with objective O3 of TayInt. Thus, with the two-step contour selection based on accuracy and precision, steps OT2-3 enforce objectives O1-2 and the kinematic cross-validation in step OT4 enforces the kinematic generalisability, O3. The lists of contour choices contained in `ChosenContourSec[k]` and `PartialChosenContourSec[k]` are those which pass all these requirements and will therefore lead to algebraic expressions which yield accurate numerical approximations that can be made more precise at arbitrary kinematic points.

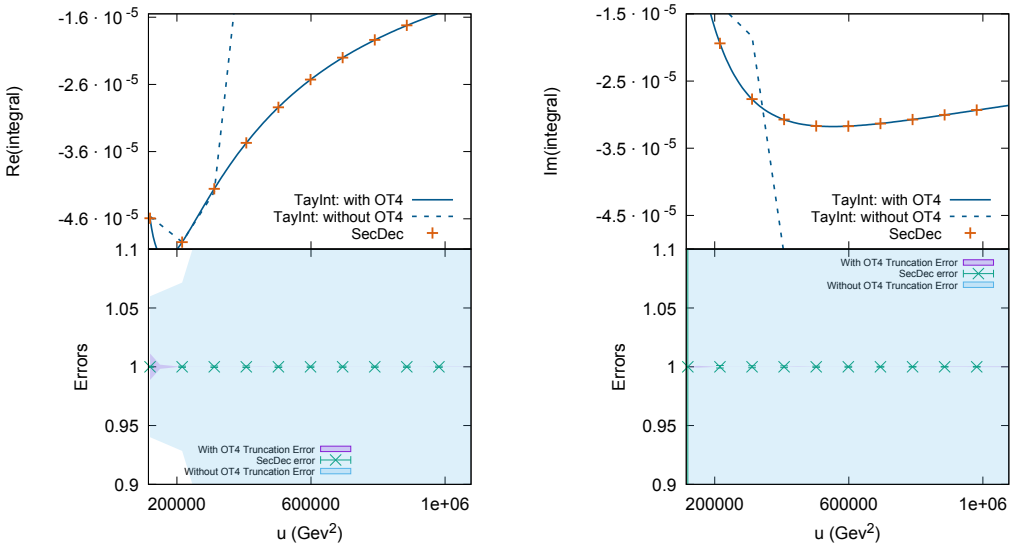


Figure 11.8: The numerical approximations for I10 at order ϵ^0 with $u \in [4m_1^2, 36m_1^2]$ and $m_h^2 = 0.5m_1^2$, $m_1 = m_t = 173$ GeV. The left panels show the real parts and the right hand panels the imaginary parts. The upper panels show the approximations produced with and without step OT4, kinematic cross-validation, while the lower panels display the associated errors. This illustrates the importance of using a kinematic training set and cross-validation sets as part of TayInt to achieve objective O3 and generate algebraic approximations that are generalisable to different kinematic points.

To see this at work, Fig. 11.8 shows the TayInt approximations for I10 at the order ϵ^0, generated using the kinematic training set which ranges from $u = 4m_1^2$ to $u = 36m_1^2$ coupled with three cross-validation sets which range from $u = 4m_1^2$ to $u = 11m_1^2$, $u = 18m_1^2$ to $u = 25m_1^2$, $u = 29m_1^2$ to $u = 36m_1^2$, alongside those generated using only a training set which ranges from $u = 4m_1^2$ to $u = 8m_1^2$, plotted over the kinematic range, $u \in [4m_1^2, 36m_1^2]$. As is made clear therein, the algebraic TayInt approximations using the larger training set and cross-validation sets in step OT4 are a better description of the Feynman integral. However, if the same algebraic approximations for I10 are then evaluated at the points in the range $u \in [4m_1^2, 8m_1^2]$, those obtained using step OT4 are once again more accurate and precise (Fig. 11.9).

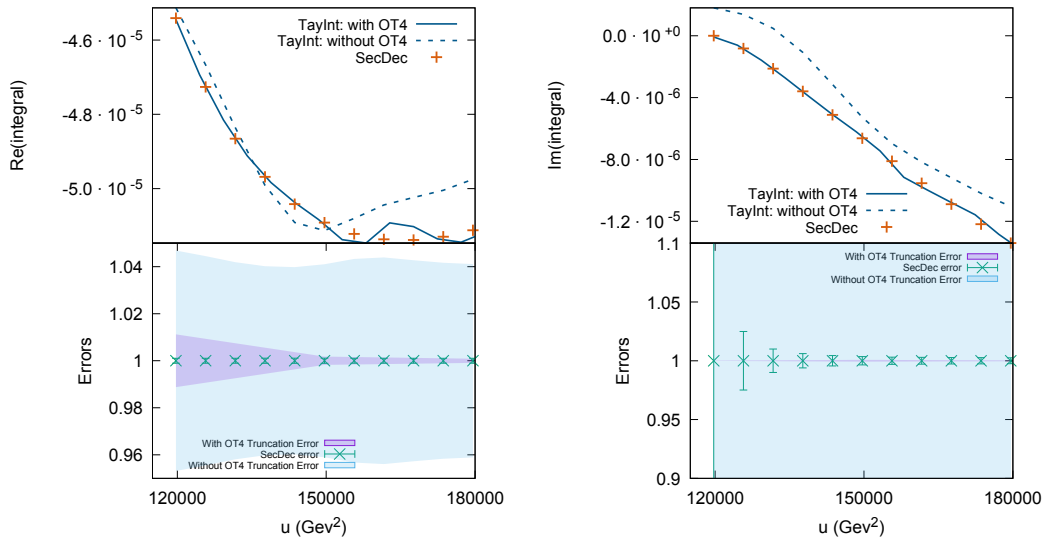


Figure 11.9: The numerical over-threshold TayInt approximations for I10 with $u \in [4m_1^2, 8m_1^2]$ and $m_h^2 = 0.5m_1^2$, $m_1 = m_t = 173$ GeV. The left panels show the real parts and the right hand panels the imaginary parts. The upper panels show the approximations produced after using the final TayInt algorithm with and without step OT4, while the lower panels display the associated errors. This reiterates the importance of using a kinematic training set and cross-validation sets within TayInt in minimising the generalisation error of the algebraic approximations for the Feynman integral and achieving objective O3.

11.8.6 L2 (OT2-4): Overview

As previously shown in Fig. 11.1, steps OT2-4 are also carried out for all the possible contour configurations in which not all of the Feynman parameters are mapped to the complex space. This means that the list of possible contour choices gains an extra dimension: the possible partial complex transformations and so is now three-dimensional rather than two-dimensional. In the case of the full contours, there was only one possible transformation, $t_j \to \frac{1}{2} - \frac{1}{2}e^{-i\theta_j}$ where j takes on values from zero to the number of Feynman parameters.

However, for the partial contours, there are several possible transformations, as j can run over the elements of any subset of the list of Feynman parameters, such as $\{0,1,2\}, \{0,1,3\}, \{0,1,4\}, \ldots$ in the case of I59. This also means that instead of having a list of integration variables,

```
varlist={theta[0],theta[1],theta[2],theta[3],theta[4]},
```
(11.55)

there is a nested list instead: `varlist` → `SubVarFinalTab`, with a different sub-list list of integration variables for each different possible transformation,

Example 11.8.6

```
SubVarFinalTab=
{{theta[0],theta[1],theta[2],t3,t4},
{theta[0],theta[1],t2,theta[3],t4},
{   ⋮   , ⋮ ,  ⋮  ,   ⋮   , ⋮ },
{t0,theta[1],t2,theta[3],theta[4]},
{t0,t1,theta[2],theta[3],theta[4]}},                    (11.56)
```

with the output in grey showing the result in the case of I59 at order ϵ^1. Finally, the list of subsectors is also promoted to a nested list, `CAdomsec[[j]]` → `CAdomsecSub[[i]][[j]]`, because each subsector takes on a different representation under the different transformations. Within this nested list, each element of each sublist (`j`) is the relevant subsector following each transformation. Therefore, the algebraic Taylor expanded and integrated approximations are generated as follows:

```
Table[COMPDiscTaySeriesIntegrator[CAdomsecSub[[#]][[i]],
    SubVarFinalTab[[i]],SubVarFinalTab[[i]],0,
    4,(ToString[name]<>"Sub"<>ToString[i]),#],{i,1,
    Length[CCNIELSub]}]&/@Range[1,Length[SecList]].
```

The partial contour configuration that gives the form of each subsector best suited to a Taylor expansion, `PartialChosenContourSec[k]`, is selected in the same way as was done in the case of the full contours, except that this time the result is a bipartite list not a number, for example:

Example 11.8.7

```
Table[PartialChosenContourSec[k],{k,1,Length[SecList]}]   (11.57)
{{5,1},{9,2},{5,3},{9,2},{1,1},{2,4},
{10,2},{10,1},{3,2},{5,4},{4,4},
{1,2},{5,4},{8,4},{9,2},{5,1},{9,2},
{9,2},{4,1},{4,1},{5,1},{5,1}},
```

in the case of I59 at order ϵ^1. The first entry denotes the transformation, the second the contour configuration on that transformation.

11.8.7 OT5: Choose Between the Optimal Full and Partial Contours

Although contours have been found which will lead to accurate, precise and generalisable approximations, it does not mean the contour has been found which is as accurate, precise and generalisable as possible. By construction, both the optimal full and partial contours are highly suited for use in the TayInt calculation. Nevertheless, in most cases one of them will more suitable than the other and in step OT5 TayInt applies quantitative methods to determine which one.

This step follows the principle of maximising information (P2). Just because a full contour can be found which is suitable for the TayInt calculation, it does not follow that the algorithm should definitely use it. The optimal partial contour may be even more conducive to the achievement of the objectives of TayInt and vice versa. Thus, the final TayInt algorithm simultaneously considers both sets of potential contours. This provides more potential for higher precision within the context of accuracy and ensures the objectives are achieved as quickly as possible. As the partial contours are smaller, the final TayInt algorithm only uses the minimal amount of complexity. Explicitly, in the case of I59 at order ϵ^1, the mean size of the fully-mapped complex subsectors is 156396 bytes, whereas the mean size of the partially-mapped complex subsectors is 103914 bytes.

However, the partial contours alone are not enough to ensure that an accurate and precise approximation for any Feynman integral can found, as sometimes a mapping of all the integration variables is strictly necessary to satisfy the objectives of TayInt. It is the union of the the full and partial contours together in final TayInt algorithm that makes it so powerful. The more precise and accurate the algorithmic analysis, the faster a precise, accurate approximation can be calculated.

Making the Comparison Valid

1. Quantitatively, the TayInt algorithm assembles a nested list of the partition:plain ratios on each optimal full and partial contour, in the order `{full,partial}` for each subsector. As described in Section 11.6 and detailed in Eqs. (11.10)-(11.13), the list of partition:plain ratios used to compare the optimal full and partial contours are chosen from the kinematic training set and second cross-validation set. This is done in such a way that the important features of each contour can be reliably assessed, without being biased by the natural deterioration in accuracy at theend points of the training set.

   This step is particularly well illustrated by the case of I10 at order ϵ^2, for which `FuParProx`, defined earlier in Eq. (11.10), reads:

Example 11.8.8

```
FuParProx=                                                    (11.58)
{{0.108,0.612,0.072,0.690,0.899,0.309,0.794,0.992} .
```

The entries of this list quantify the relative deviation between the man absolute partition:plain ratios over the kinematic training set for the full and partial contours. The mean absolute full-partial deviation over the training set for subsectors 1 and 3 falls below the threshold of 0.15 (defined in Section 11.6 and summarised in Table 11.2) necessary for the algorithm to distinguish the precision and accuracy associated with each contour. This means that the larger partition:plain ratios at the end points of the kinematic training set may overshadow the deviations between the partition:plain ratios that are a consequence of the contour, leading the algorithm to focus instead on the deviations that arise due to the extent of the breakdown of the integrand at the kinematic end points. So, for these subsectors, the partition:plain ratios from the second cross-validation set are used to compare the optimal full and partial contours (as explained in Section 11.6). In the case of I10 at order ϵ^2, this reads:

Example 11.8.9

```
CrossFuParComp=                                          (11.59)
{{1.286,1.071,1.176},
{1.390,0.919,0.969}}
{{ ⋮ , ⋮ , ⋮ },
{ ⋮ , ⋮ , ⋮ }}
{{1.023,1.068,1.171},
{1.358,0.264,1.122}} .
```

Referring to Eq. (11.58), there is a clear enough difference between the optimal full and partial contours to allow the final TayInt algorithm to reliably judge their accuracy rather than their breakdown in the case of subsectors 2, 4, 5, 6, 7 and 8. Thus, the comparison of full and partial partition:plain ratios is constructed using the kinematic training set. In the case of I10 at order ϵ^2, this reads:

Example 11.8.10

```
TrainFuParComp=                                                    (11.60)
{{1.131,1.303,1.088},
{1.157,1.001,0.984}}
{{ ⋮ , ⋮ , ⋮ },
{ ⋮ , ⋮ , ⋮ }}
{{1.052,1.108,1.209},
{401.888,0.631,0.607}} .
```

The virtue of accounting for the kinematic end points can be seen in the case of I10 subsector 1 at order ϵ^2, see Fig. 11.10. The TayInt approximation, with and without incorporating the cross-validation and training set into the full vs. partial contour decision, is displayed in this plot. As this illustrates, failing to incorporate the second cross-validation set leads to an approximation which is accurate and precise at the beginning and end of the kinematic region but breaks down in the middle. The deviation between the full and partial contour in the kinematic bulk is on a smaller scale than the deviation at the end points. Thus it cannot be seen using the mean absolute partition:plain ratios over the training set, which is biased by the lager end point ratios. Therefore, using the ratios generated with the training set for subsector 1 of I10 at order ϵ^2 leads to the contour which minimises the breakdown at the end points rather than that which maximises the accuracy and precision of the contour itself being chosen.

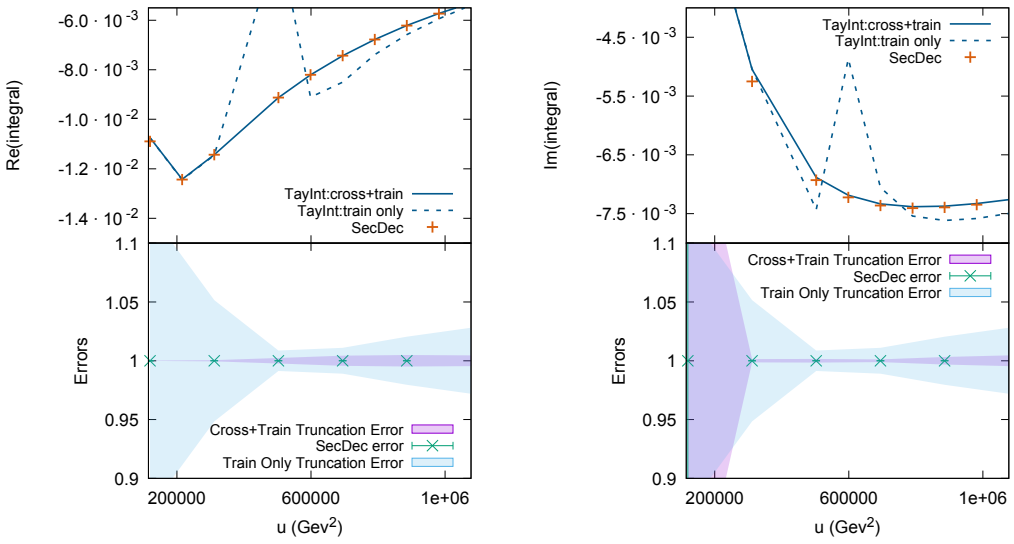


Figure 11.10: The numerical over-threshold approximations for I10 at order ϵ^2 with $u \in [4m_1^2, 36m_1^2]$ and $m_h^2 = 0.5m_1^2$, $m_1 = m_t = 173$ GeV. The left panels show the real parts and the right hand panels the imaginary parts. The upper panels show the approximations produced by TayInt with its usual cross-validation and training set method of choosing between the full and partial contours and those produced by TayInt if only the training set based ratios are used. The lower panels display the associated errors. This illustrates the importance of incorporating both the training set and second cross-validation set into the decision between the optimal full and partial contour.

The I10 integral at order ϵ^2 was particularly illustrative of the method of setting up the comparison list, Eq. (11.60). However the process of choosing between the optimal full and partial contour based on this list is best represented by the case of I59 at order ϵ^1, for which the comparison list reads:

Example 11.8.11

```
FuParComp=
{{0.923,0.420,1.094,1.027,0.939},
{1.179,1.029,0.887,0.425,0.710}}
{{ ⋮ , ⋮ , ⋮ , ⋮ , ⋮ },
{ ⋮ , ⋮ , ⋮ , ⋮ , ⋮ }}
{{0.665,0.888,1.061,0.581,1.160},
{0.782,1.044,0.738,1.020,0.824}} .
```
(11.61)

At the end of this part of the final TayInt algorithm a set of numbers for the optimal full and partial contours has been assembled which facilitates a valid comparison. In what follows, that comparison will be made and used to pick the final chosen contour configuration for each subsector of I59 at order ϵ^1.

Negative Exclusion

2. The properties of each sub-list in Eq. (11.61) are first assessed according to the principle of negative exclusion (P3). The final TayInt algorithm counts the number of variables on the optimal full and partial contours which have partition:plain ratios in the range $[0.5, 1.25]$, returning a bipartite list for each subsector, as follows:

Example 11.8.12

```
Table[NumBadVarSec[k]=                                    (11.62)
{{0,0},{0,0},{1,1},{1,1},{1,0}
{0,0},{4,0},{1,0},{1,0},{0,0},
{1,0},{4,1},{1,0},{0,0},{0,0}},
{0,0},{0,0},{0,0},{0,0},{0,0},{1,0},{0,0}} .
```

The grey output in Eq. (11.62) shows the result in the case of I59 at order ϵ^1, in which instance, the partial contour is chosen for the subsectors $5, 7, 8, 9, 11, 12, 13, 21$ at this stage. This is because the optimal full contour for those subsectors contains an integration variable with a partition:plain ratio in the range $[0.5, 1.25]$ but the optimal partial contour contains one or fewer. This is encoded by assembling a list of 1s and 2s indicating that the full or partial contour has been chosen, or a list `{1,2}` if there are an equal number of unsuitable ratios, as follows:

Example 11.8.13

```
BaCho=(Flatten[Position[NumBadVarSec[#],
      Min[NumBadVarSec[#]]]])
      &/@Range[1,Length[SecList]]                    (11.63)
      {{1,2},{1,2},{1,2},{1,2},{2}
      {1,2},{2},{2},{2},{1,2},
      {2},{2},{2},{1,2},{1,2}},
      {1,2},{1,2},{1,2},{1,2},{1,2},{2},{1,2}} .
```

If there is an equal non-zero number of unsuitable variables in the chosen full and partial contours, then the mean partition:plain ratio, over the set of those variables, is calculated:

Example 11.8.14

```
Table[If[Length[BaDiffCho[[k]]]==2,                  (11.64)
ActHighBadVarSec[k]=
Table[Select[FuParComp[[k]][[i]],#>1.25&][[1]],
{i,1,Length[FuParComp[[k]]]}],{k,1,Length[SecList]}]
{Null, Null,{1.52102,1.25439},{2.36699,1.33688},Null
Null,Null,Null,Null,Null,
Null,Null,Null,Null,Null,
Null,Null,Null,Null,Null,Null,Null} .
```

If not, a `Null` entry is returned and the contour with the lower such mean is selected. In this way, the partial contour is chosen for subsectors 3 and 4, as can be seen from Eq. (11.64). `BaCho` is then updated, so that for I59 at order ϵ^1 it now reads:

Example 11.8.15

```
BaCho=(Flatten[Position[NumBadVarSec[#],
      Min[NumBadVarSec[#]]]]) 
      &/@Range[1,Length[SecList]]                    (11.65)
      {{1,2},{1,2},{2},{2},{2}
      {1,2},{2},{2},{2},{1,2},
      {2},{2},{2},{1,2},{1,2}},
      {1,2},{1,2},{1,2},{1,2},{1,2},{2},{1,2}} .
```

Next, the real and imaginary parts of the partition:plain ratios are subject to the same classification. However this does not add any new information in the case of I59 at order ϵ^1, so no further selection is made. The TayInt algorithm moves on to test the accuracy, of only those subsectors for which the negative exclusion test, Eq. (11.65), was indecisive, returning `{1,2}`.

Accuracy

3. Next, the accuracy of the optimal full and partial contours is tested by counting the number of integration variables with partition:plain ratios in the range $[0.8, 1.2]$. In the case of I59 at order ϵ^1, this leads to the nested list:

Example 11.8.16

```
Table[NumCompVarSec[k]=                              (11.66)
{4,3},{4,5},{4,3},{3,2},{2,4}
{4,5},{1,5},{4,4},{1,3},{3,5},
{2,5},{1,4},{3,4},{4,3},{5,4}},
{4,3},{4,4},{4,2},{5,3},{2,4},{3,4},{3,3}},
```

and likewise for the ratios in the real and imaginary parts, which must also be checked in order for a contour to be selected in this way. If the same contour has the highest number of integration variables with partition:plain ratios in `FuParComp` suggestive of accuracy in the absolute value, real and imaginary parts, then it is selected from the bipartite lists in Eq. (11.66). If the verdict is not unanimous, then no selection is made at this stage. It is very important to check the imaginary part, because it frequently behaves very differently to the real part. But, this would not always be detected by simply assessing the ratios of the absolute values. Thus, in the case of I59 at order ϵ^1, the full contour is chosen for subsectors $1, 14, 15, 16, 18$

and 19 and the partial contour is selected for subsectors 2, 6, 10 and 20, as can be seen by referring to Eq. (11.66). This is encoded by assembling a list of 1s and 2s indicating that the full or partial contour has been chosen, or a list `{1,2}` if there are an equal number of accurate ratios, as follows:

Example 11.8.17

```
CompCho=(Flatten[Position[NumCompVarSec[#],                    (11.67)
        Max[NumCompVarSec[#]]]])
        &/@Range[1,Length[SecList]]                            (11.68)
        {{1},{2},{1},{1},{2}
        {2},{2},{1,2},{2},{2},
        {2},{2},{2},{1},{1}},
        {1},{1,2},{1},{1},{2},{2},{1,2}} .
```

If the accuracy check is indecisive and Eqq. (11.66) returns a list of equal numbers for certain subsectors, then the TayInt algorithm moves on to the final test of the optimal full and partial contours. In accordance with objectives O1-2 of the final TayInt algorithm, this is the improvement check.

Improvement

4. If the optimal full and partial contours have not been differentiated by negative exclusion or by accuracy, then the difference between them is not so substantial as can be seen by checking accuracy. Therefore, the potential for improvement is assessed, quantified by the difference between the number of integration variables which have partition:plain ratios in the range $[0.5, 0.7]$ (improving) and those in the range $[0, 0.3] \cup [1.25, \infty)$ (unsuitable). The TayInt algorithm first finds the number of unsuitable variables per contour for each subsector:

```
Table[NumBadVarSec[k]=Table[Length[Join[Flatten[Position[
FuParComp[[k]][[i]],_?(0<#<0.3&)]],
Flatten[Position[FuParComp[[k]][[i]],_?(#>1.25&)]]]],
{i,1,Length[FuParComp[[k]]]}],{k,1,Length[SecList]}] .          (11.69)
```

It then counts the number of variables with partition:plain ratios indicative of improvement:

```
Table[NumVeryGoodVarSec[k]=Table[Length[Flatten[Position[
FuParComp[[k]][[i]],_?(0.5<#<0.7&)]]],
i,1,Length[FuParComp[[k]]]],{k,1,Length[SecList]}] ,            (11.70)
```

computes the difference between the two numbers per contour for each subsector;

```
GoodBaDiff=Table[
-NumBadVarSec[k] + NumVeryGoodVarSec[k],
{k,1,Length[SecList]}] ,                                    (11.71)
```

before returning the position within each sub-list of the maximum;

```
GoodBaDiffCho=
(Flatten[Position[GoodBaDiff[#],Max[GoodBaDiff[#]]]])
&/@Range[1,Length[SecList]] .                               (11.72)
```

In the case of I59 at order ϵ^1 this reads:

Example 11.8.18

```
GoodBaDiffCho=                                              (11.73)
{{1,2},{1,2},{2},{1,2},{1,2}
{1,2},{2},{2},{1,2},{1},
{2},{2},{2},{2},{2}},
{1},{1,2},{1,2},{1,2},{1},{1,2},{1}} .
```

As can be seen from the grey output in Eq. (11.73), in the case of I59 at order ϵ^1, subsector 22 is assigned a full contour after this step.

Finally, the partial contour is selected by default because there is no merit in paying the additional computational price for choosing a full contour if using it makes no difference to accuracy or improvement. At this stage all the subsectors of I59 at order ϵ^1 except $2, 3, 6, 14$ and 17 have been allocated a full or partial contour, so in these cases the optimal partial contour is chosen. The ultimate outcome of this part of the code is to return a list of numbers, either 1 (denoting the full contour) or 2 (denoting the partial contour) for each subsector. This is done by taking the entries of `BaCho`, Eq. (11.65), `CompCho`, Eq. (11.68) and `GoodBaDiffCho`, Eq. (11.73), in that order and taking a 2 for all those unassigned. In the case of I59 at order ϵ^1, this reads:

Example 11.8.19

```
FuParDecCho=                                                (11.74)
{2,2,2,2,2,2,2,2,2,2,2,2,
2,2,1,1,1,2,1,1,2,2,1}.                                     (11.75)
```

Based on this, the final list of contours can be assembled from the full and partial formats, using the code:

```
CCFinal=Which[FuParDecCho[[#]]==1,
ChosenContourSec[#],FuParDecCho[[#]]==2,
PartialChosenContourSec[#]]&/@Range[1, Length[SecList]],
```
(11.76)

which in the case of I59 at order ϵ^1 reads:

Example 11.8.20

```
CCFinal=
{{5,1},{9,2},{5,3},{9,2},{10,3},{2,4},{10,2},{10,1},{3,2},
{5,4},{4,4},{8,3},{5,4},16,11,3,{9,2},18,4,{8,3},{5,1},1} .
```
(11.77)

The single numbers represent the position of the optimal partial contour in `CCNIELcalc`, whereas the bipartite lists specify the position of the optimal partial contour in `CCNIELpartialcalc`.

11.8.8 Contour Choice Summary: OT2-5

The decision-making process in OT5 is summarised in Table 11.3 in the case of I59 at order ϵ^1 and its merit portrayed by Fig. 11.11. This plot shows the improvement obtained by using the final TayInt algorithm rather than the conceptual algorithm for subsector 15 of I59 at order ϵ^1. This clearly shows the necessity of implementing changes C1-3 to achieve objectives O1-3 for more complicated Feynman integrals.

Table 11.3: The breakdown of how the full and partial contours are selected for the 22 subsectors of I59 at order ϵ^1. The numbers indicate the subsector and their presence in either the full or partial row conveys that the full or partial contour was chosen for those subsectors. If a criterion confirms choices that have already been made, hence leads to no new information, the corresponding column entry remains empty. The partial contour is used for subsectors $2, 3, 6, 14$ and 17 by default as the various criteria cannot decide between the full and partial contours for this subsectors.

Contour	Bad	Accurate	Difference	Default
Full	–	$\{1, 15, 16, 18, 19\}$	$\{22\}$	–
Partial	$\{4, 5, 7, 8, 9, 11, 12, 13, 21\}$	$\{10, 20\}$	–	$\{2, 3, 6, 14, 17\}$

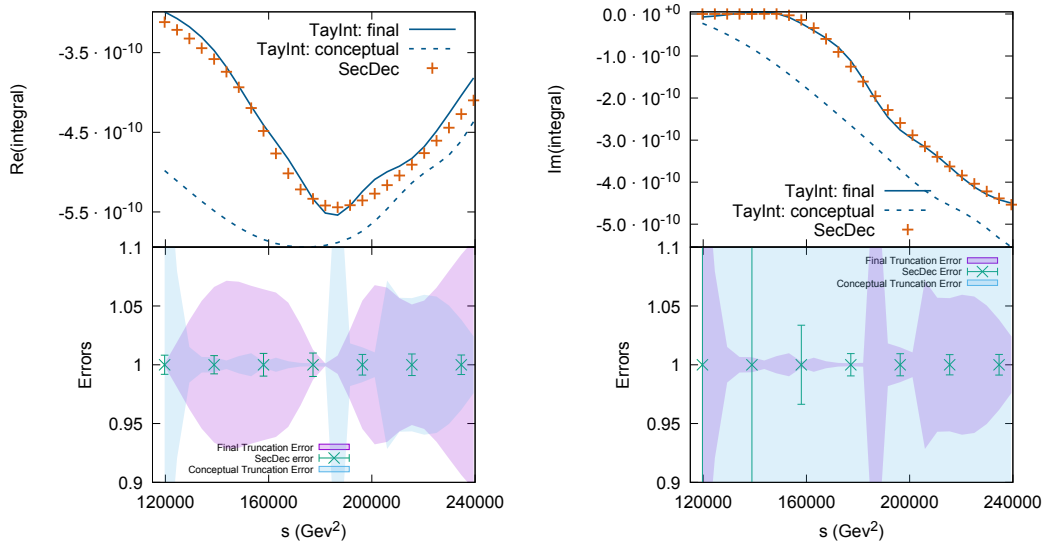
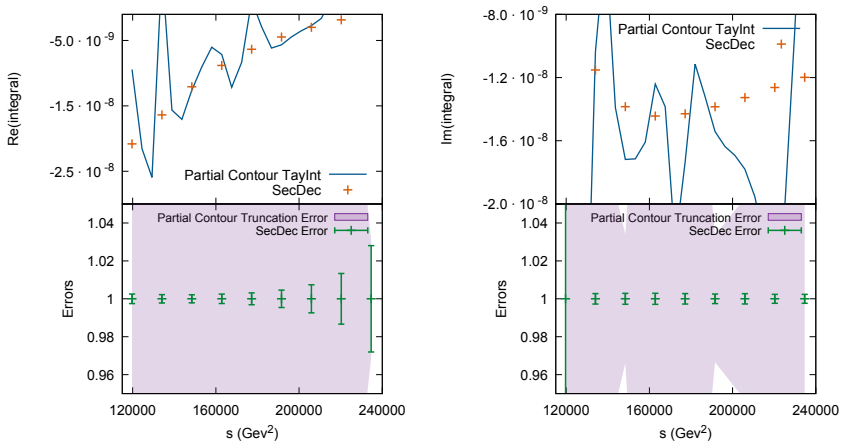
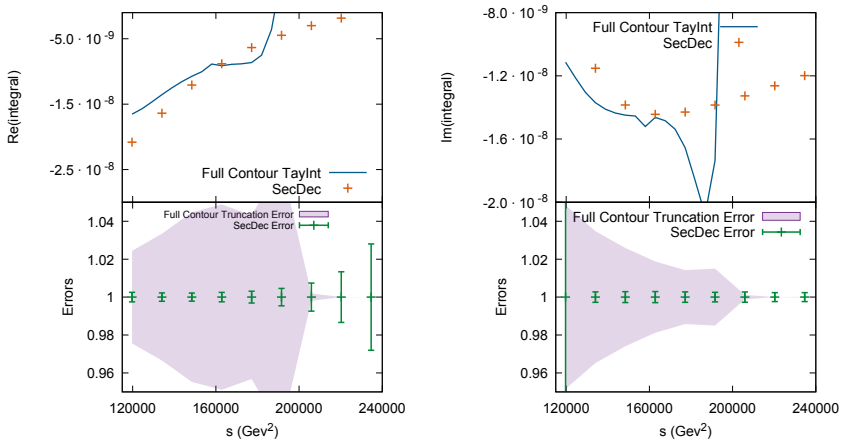


Figure 11.11: The numerical over-threshold approximations (with a fourth order series expansion) for subsector 15 of I59 with $s > 4m_1^2$ and $u = -2m_1^2$, $m_h^2 = 0.5m_1^2$, $m_1 = m_t = 173$ GeV. The left panels show the real parts and the right hand panels the imaginary parts. The upper panels show the approximations produced by the conceptual and final TayInt algorithm, whereas the lower panels display the associated errors. This illustrates the improvement achieved by the final version of TayInt due to changes C1-3.

To further and more completely demonstrate the virtue of the changes C1-3 implemented in the final version of TayInt, the full approximations for the I59 integral at order ϵ^1 over threshold are displayed in Figs. 11.12 and 11.13. In the former, the approximations using the partial contours exclusively (L2) and the full contours exclusively (L1) are portrayed. In the latter, the approximations using the conceptual TayInt algorithm are compared to the approximations using the final TayInt algorithm, which maximises information by incorporating both full and partial contours. The important difference between the two approximations is that the latter converges much faster in the bulk of the kinematic region, away from the threshold, whereas the former exhibits deteriorating convergence.

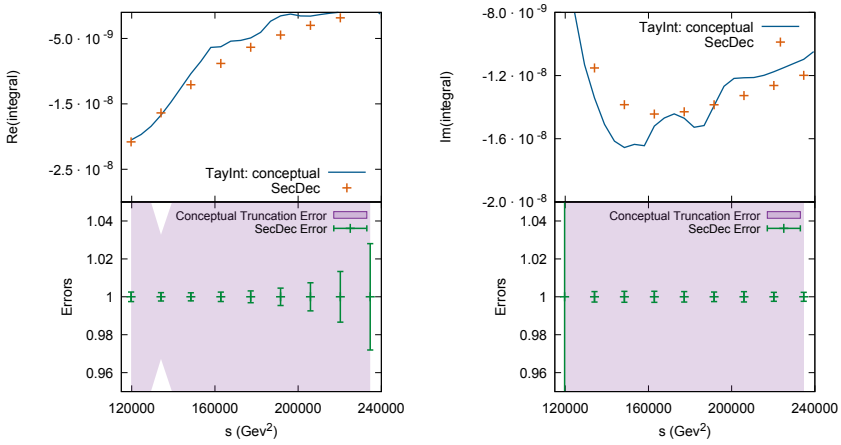


(a)

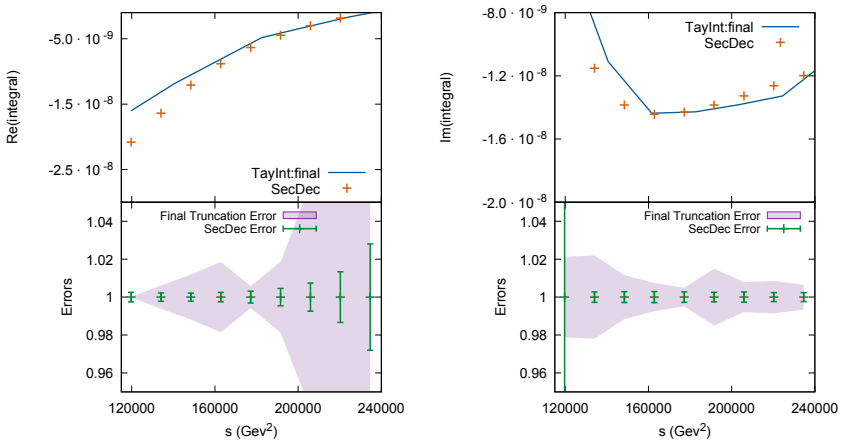


(b)

Figure 11.12: I59 at order ϵ^1 over threshold, calculated with a fourth order series expansion using; (a) the partial contours exclusively, (b) the full contours exclusively, with $t = -2m_t^2, m_h^2 = 0.5m_t^2$ and s running from 0 to $8m_t^2$, where $m_t^2 = 29929$ GeV^2, the top mass squared. The lower plots show the relative uncertainties of the TayInt approximations and the corresponding SecDecv3 uncertainties.



(a)



(b)

Figure 11.13: I59 at order ϵ^1 over threshold, calculated with a fourth order series expansion using; (a) the conceptual TayInt algorithm and (b) the final TayInt algorithm, with $t = -2m_t^2, m_h^2 = 0.5m_t^2$ and s running from 0 to $8m_t^2$, where $m_t^2 = 29929$ GeV^2, the top mass squared. The lower plots show the relative uncertainties of the TayInt approximations and the corresponding SecDecv3 uncertainties.

11.8.9 L3 (OT6): Generate the Uniform Partition Ratios on the Chosen Contours

Change C1 has now been implemented and the contour configurations that best achieve objectives O1-3 of accuracy, improvement and kinematic generalisability have been chosen for each subsector. However, to take the most advantage of the potential for improvement of the chosen contours, the optimal way to partition them must be found (C2). As explained in Section 11.6, there are three possibilities, high-uniform, low-uniform, or varied partition sets. In order to quantitatively decide which to apply to each subsector, in step OT6 the ratios defined in Eqs. (11.14)-(11.18) are calculated, which, in the case of I59 at order ϵ^1 read:

Example 11.8.21

```
Hom3PlainAbsRat=                                                    (11.78)
{0.604,0.852,0.779,16.274,0.900,1.013,0.900,
0.741,0.779,0.946,1.029,1.187,0.901,0.638,
0.858,0.564,0.583,0.428,0.760,0.723,0.758,0.250},
```

Example 11.8.22

```
Hom3PlainReRat=                                                     (11.79)
{5.539,1.021,4.209,14.900,1.152,1.116,0.700,
0.825,1.062,1.086,1.252,2.500,0.766,0.532,
1.014,1.292,0.429,0.399,1.367,0.587,6.321,0.204},
```

Example 11.8.23

```
Hom3PlainImRat=                                                     (11.80)
{0.723,0.752,0.584,20.164,0.323,0.518,1.417,
0.671,0.274,0.296,0.483,1.325,0.890,0.634,
1.545,0.469,1.525,0.797,0.917,3.258,0.555,0.228},
```

Example 11.8.24

```
Hom2PlainAbsRat=                                                    (11.81)
{0.581,0.848,0.825,1.901,0.914,1.020,0.941,
0.713,0.814,0.949,1.028,1.157,0.986,0.653,
0.882,0.512,0.578,0.414,0.702,0.659,0.767,0.174},
```

Example 11.8.25

```
Hom3Hom2AbsRat=                                                     (11.82)
{1.187,1.004,0.939,7.971,0.980,0.990,0.966,
1.041,0.955,0.986,0.983,1.016,0.948,1.033,
1.015,1.110,2.485,1.098,1.096,2.034,0.982,1.480},
```

At the end of step OT6, the final TayInt algorithm has computed the ratios which can be used to numerically determine which set of partitions is best suited to improving the precision and accuracy of each chosen contour in `CCFinal`. This implements change C2 that allows the final version of TayInt to tackle more complicated elliptic integrals, such as I59.

11.8.10 L3 (OT7): Assess the Probable Accuracy of the Chosen Contours

The next step, OT7, in using the above ratios to choose the partition sets for each subsector is to assess whether or not the chosen contours are alone sufficient to guarantee the accuracy of the approximation produced using them, using the mimicry principle (P1). As explained in Section 11.6, the mean of `Hom3Hom2AbsRat[k]` (defined in Eq. (11.18), illustrated in Eq. (11.82)) over the subsectors `k` is computed. If it is in the range $[0.95, 1.05]$ then a high-uniform partition set is used for each subsector. No further analysis is carried out. This is the case for the integrals I10 and I39 at all orders in ϵ. These Feynman integrals are 'simple' enough for the chosen contour to already guarantee a very accurate approximation. However, for I59 at order ϵ^1, a highly complicated integral, this mean is 1.468. This number informs the TayInt algorithm that the chosen contours are not enough to guarantee sufficient accuracy in the ensuing calculation, due to the complicated form of the subsector integrands. Thus, the partition set must also be chosen to improve the accuracy of the subsequent approximation. At the end of the step OT7, the final TayInt algorithm has established that a high-uniform partitioning cannot be applied to all of the subsectors of I59 at order ϵ^1, as would be the case for I10 and I39.

11.8.11 L3 (OT8): Assess the Suitability of Low- or High-Uniform Partition Sets

As the mean of `Hom3Hom2AbsRat[k]` over the subsectors `k` of I59 at order ϵ^1 is outside the range $[0.95, 1.05]$, then the final TayInt algorithm proceeds to step OT8. The goal of this step is to ascertain which of the subsectors can be calculated accurately with a high-uniform partition set and if a low-uniform partition set is sufficient to accurately calculate the remainder. Applying the criteria of accuracy and improvement given in Section 11.6 leads to the following classifications.

Accuracy

None of the subsectors of I59 at order ϵ^1, are sufficiently accurate to be subsequently calculated with a high-uniform partition set.

Improvement

In the case of I59 at order ϵ^1 the subsectors 1, 16, 17 and 18 are improving quickly enough to be calculated with a high-uniform partition set.

Lack of Accuracy and Improvement

Referring to the ratios in Eqs. (11.78)-(11.82), the uniform partitioning lacks both accuracy and improvement for subsector 4 of I59 at order ϵ^1. In this case, the individual variables must be separately analysed in step OT9 to construct a set of partitions that can coax an accurate approximation for the subsector integral out of its form on the contour chosen for it in steps OT2-5.

Remainder

A partitioning of `{1,1,1}` is used in each variable of subsectors 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 19, 20, 21 and 22 of I59 at order ϵ^1, whose ratios in Eqs. (11.78)-(11.82) do not fall into any of the above classes. This is a remarkably conservative decision boundary, in accordance with the safety-first principle of the TayInt algorithm (P4). In the case of I59 at order ϵ^1, the approximations for subsectors 1, 5, 6, 8, 9, 10, 14, 15, 16, 17, 18, 19, 20 and 22 are most precise and accurate when a high-uniform partitioning is used. However, the final TayInt algorithm only selects a high partitioning for subsectors 1, 16, 17 and 18, using a low-uniform partitioning for the remainder. If a less stringent decision boundary was used, such as only demanding that `0.9<Hom3PlainAbsRat<1.1` rather than `0.95<Hom3PlainAbsRat<1.05`, `0.95<Hom2PlainAbsRat<1.05`, `0.95<Hom3PlainReRat<1.05` and `0.95<Hom3PlainImRat<1.05`, then a high-uniform partitioning would be assigned to subsectors 6, 11, 19 and 20 using the accuracy criterion. This would generate more precise approximations for subsectors 6 and 19. But, the over-partitioning of subsector 11 would lead to a much

more inaccurate approximation for that subsector, for which the most precise and accurate approximation is generated with a low-uniform partitioning. Thus, the overall approximation for I59 would be far less accurate, despite the optimal approximation being generated for more of the subsectors.

To reiterate, it is much more important to make sure that the approximations for *all* the subsectors are precise and accurate, rather than maximising the number of subsectors with the most precise and accurate approximations at the expense of producing an inaccurate approximation for one subsector. Thus it is crucial that TayInt only classifies subsectors which exceed the minimal decision boundary by a large margin, in accordance with the safety-first principle (P4). Of course, the optimal approximations for I59 at order ϵ^1 could also be produced by using a very detailed decision-making process which is tuned to fit the I59 integral exactly. But this would undermine the general applicability of TayInt: using a very intricate decision-making process to choose the partitions which yield the optimal approximations for a particular integral has a high generalisation error. If TayInt was written to produce the optimal approximations for I59, its decision-making process would be overfitted to that integral and would make decisions that lead to inaccurate approximations for other different integrals. Minimising the generalisation error is one of the core objectives (O3) of TayInt, as was also seen in step OT4 when selecting the contours with low kinematic generalisation error. At the end of step OT8, all the subsectors of I59 at order ϵ^1 except subsector 4 have been assigned a partition set.

11.8.12 L4 (OT9): Determine the Varied Partitioning for those Subsectors Requiring it

Therefore, by the time step OT9 is reached, only subsector 4 of I59 at order ϵ^1 requires further analysis. In step OT8 it was established that this subsector has neither accuracy nor improvement when calculated with a uniform partition set, see Eqs. (11.78)-(11.82). Therefore, to find an accurate representation for it, each individual variable must be partitioned differently, leading to a varied partitioning.

If a subsector has an integrand that varies considerably in a particular variable, then by using more pieces a Taylor expansion will better represent it and generate a more precise approximation. However, if the subsector is flat in a given variable, then dividing the Taylor expansion into smaller pieces resolves detail that is not there, leading to an approximation that is less accurate. In order to scrutinise the subsectors on the level of each individual integration variable, the list of partitions specified in Eq. (11.20) is used to generate approximations for each subsector. These correspond to each variable of integration being partitioned twice, thrice and four times, on the chosen contour of integration.

As explained in Section 11.6, because it is possible for a subsector to have a rapidly varying real part but a flat imaginary part, a set of partitions is chosen for both the real and imaginary parts of each subsector. Using the approximations generated with `ParTest` (Eq. (11.20)) the final TayInt algorithm then produces bipartite lists, `PartReRat` and `PartImRat`. These contain ratios, for each integration variable, of

the four-partition and three-partition approximation and the three-partition and two-partition approximation, for the necessary subsectors. Those variables of integration which are well suited to a high number of partitions are identified by their having differences between the quartic to triple ratio and the triple to double ratio less than -0.1. The subsectors well suited to a low number of partitions are identified by their having such differences greater than 0. In the case of an intermediate difference, a correspondingly intermediate number of partitions, `{1,1,1}`, is used for the relevant integration variable. For the fourth subsector of I59 at order ϵ^1, the chosen varied partition sets read:

Example 11.8.26

```
ChosenPartReSec[4]=
{{{1,1,1},{1,1,1,1,1,1},{1,1,1,1,1,1},{1},{1,1,1,1,1,1}},
```

(11.83)

```
ChosenPartImSec[4]=
{{{1},{1},{1,1,1,1,1,1},{1},{1,1,1,1,1,1}}},
```

(11.84)

following the logic encapsulated by the Eqs. (11.21)-(11.26) in Section 11.6. At the end of this step, the partition sets have been chosen for every subsector of I59 at order ϵ^1, meaning that the final TayInt algorithm has finished. It finally stores the relevant information, which will be loaded TayInt calculation code for use in producing algebraic and subsequent numerical approximations.

11.8.13 Partition Choice Summary: OT6-9

The necessity of using both uniform and precision partitions together is illustrated in Fig. 11.14. This depicts the TayInt approximations for subsector 4 of I59 at order ϵ^1. As previously shown (Eqs. (11.83)-(11.84)), TayInt uses a varied partition set for this subsector. To underline the importance of this choice, the approximations obtained using a uniform partition set for this subsector are also plotted. On the other hand, it is equally important to demonstrate that the varied partitioning is not always applicable. If mistakenly applied to a highly complicated subsector which requires a uniform partition set, loss of accuracy will ensue and imprecise, fluctuating approximations will be generated. This is depicted in Fig. 11.15, which contains the approximations for subsector 11 of I59 at order ϵ^1 obtained using a varied partition set and a uniform partition set.

In summary, it is of vital importance that both uniform and varied types of partition are considered and the one most indicative of accurate and precise approximations for each subsector chosen, if such approximations for highly complicated Feynman integrals are to be produced at all kinematic points.

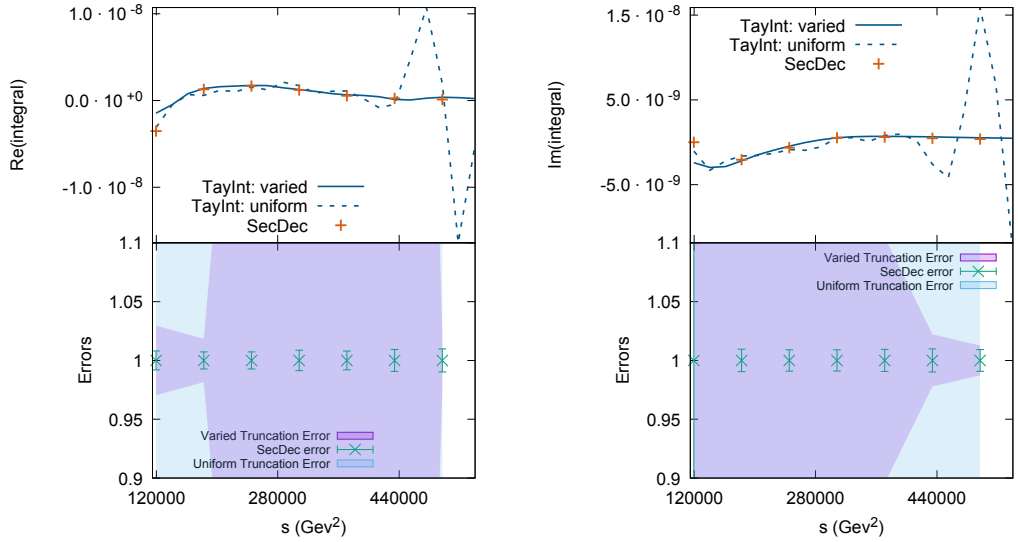


Figure 11.14: The numerical over-threshold approximations for subsector 4 of I59 with $s > 4m_1^2$ and $u = -2m_1^2$, $m_h^2 = 0.5m_1^2$, $m_1 = m_t = 173$ GeV, using the final TayInt algorithm. The left panels show the real parts and the right hand panels the imaginary parts. The upper panels show the approximations produced by the final TayInt algorithm, using varied and uniform partition sets, whereas the lower panels display the associated errors. This illustrates how combining the varied and uniform partition sets in the final TayInt algorithm, leads to the implementation of change C2 and acheives objective O2.

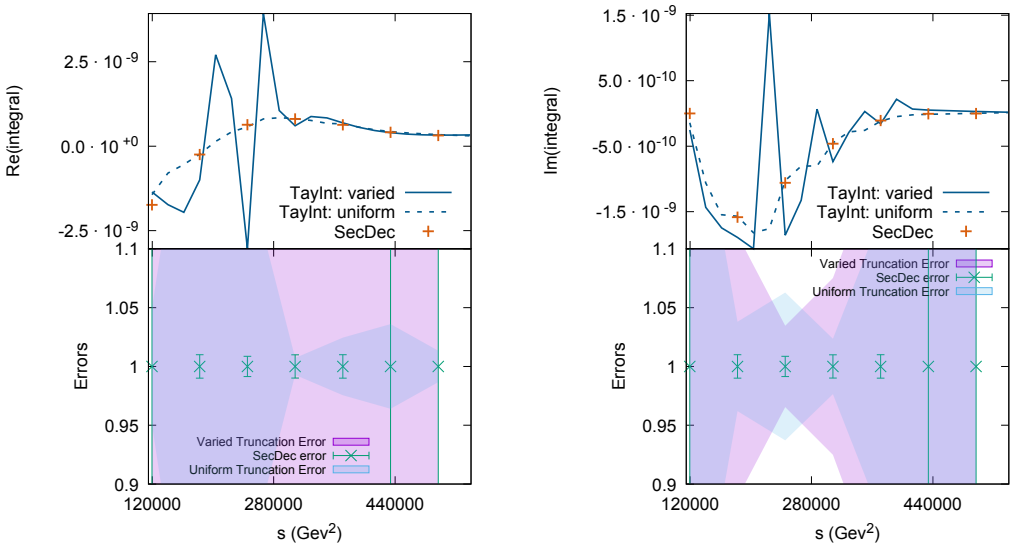


Figure 11.15: The numerical over-threshold approximations for subsector 11 of I59 with $s > 4m_1^2$ and $u = -2m_1^2$, $m_h^2 = 0.5m_1^2$, $m_1 = m_t = 173$ GeV. The left panels show the real parts and the right hand panels the imaginary parts. The upper panels show the approximations produced by the final TayInt algorithm, using varied and uniform partition sets, whereas the lower panels display the associated errors. This illustrates the necessity of correctly choosing the partition sets for each subsector in the final version of TayInt with steps OT6-9, when implementing change C2 to realise objective O2.

11.8.14 OT10: Produce the Systematic Approximation

A final demonstration of how the final TayInt algorithm, when all the pieces OT1-9 are used synchronously, achieves the objectives O1-3 for a highly complicated elliptic integral will now be given. The numerical approximations for the elliptic I59 integral at order ϵ^1 are presented over two kinematic ranges, $s \in [4m_t^2, 8m_t^2]$ and $s \in [4m_t^2, 18m_t^2]$, in Figs. 11.16 and 11.17. This Feynman integral is riddled with turning points, thus is maximally difficult to approximate using a Taylor expansion, underlining the power of the final TayInt algorithm. Moreover, both sets of numerical approximations are generated from the same systematic algebraic approximation. This emphasises how the algebraic TayInt approximations generalise to different kinematic regions (O3), producing accurate (O1) approximations whose precision is controlled by partitions (O2) even for integrals full of turning points, after making the changes C1-3. As can be seen, for the kinematic region with $s > 10m_t^2$, the I59 integral at order ϵ^1 is very difficult to describe using a Taylor expansion and it is here that steps OT6-9 are also needed to produce an accurate approximation.

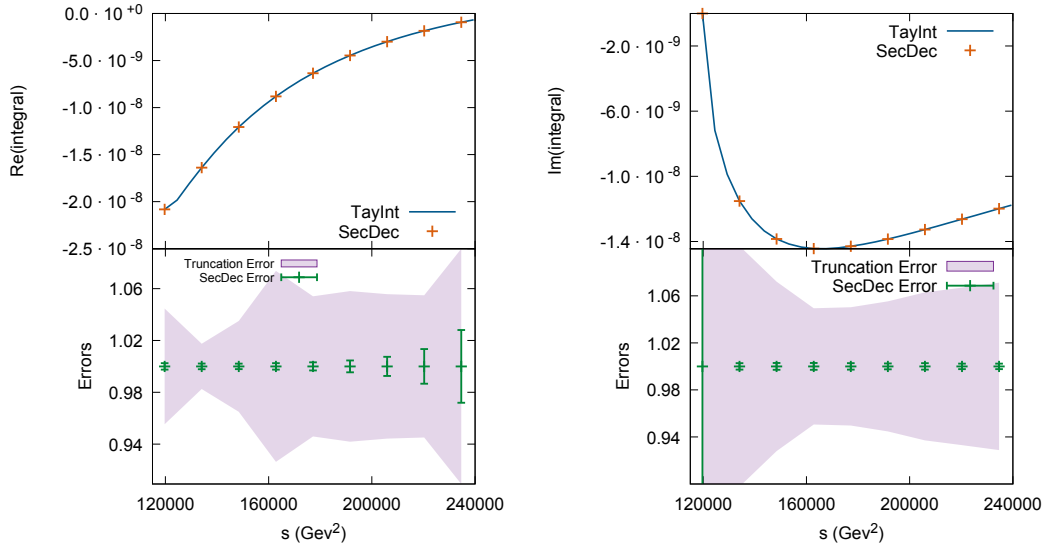


Figure 11.16: The numerical over-threshold TayInt approximations for I59 at order ϵ^1 with $s \in [4m_1^2, 8m_1^2]$ and $m_h^2 = 0.5m_1^2$, $m_1 = m_t = 173$ GeV. The left panels show the real parts and the right hand panels the imaginary parts. The upper panels show the approximations produced after using the final TayInt algorithm and by SecDec, while the lower panels display the associated errors. This integral contains many turning points and thus is maximally difficult to describe using a Taylor expansion.

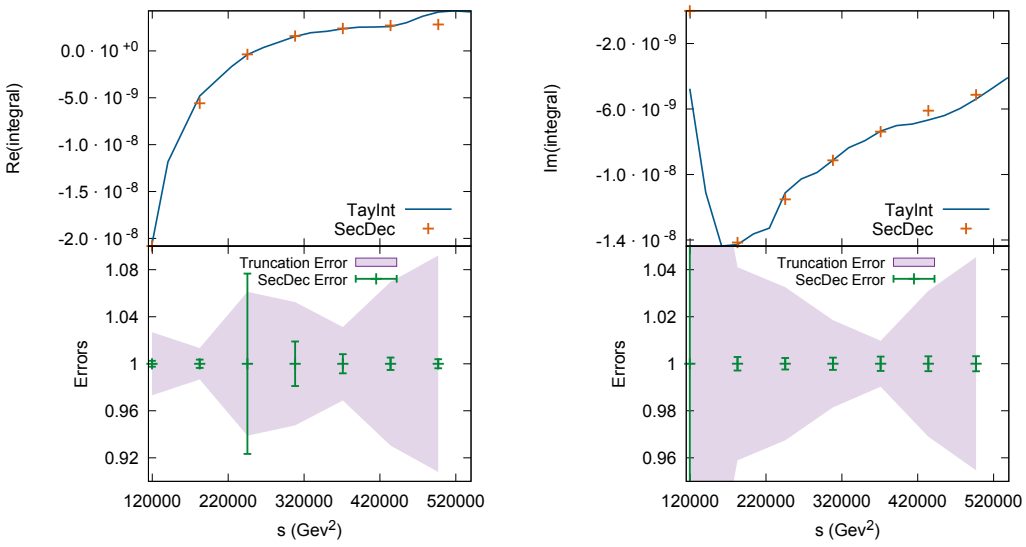


Figure 11.17: The numerical over-threshold TayInt approximations for I59 at order ϵ^1 with $s \in [4m_1^2, 18m_1^2]$ and $m_h^2 = 0.5m_1^2$, $m_1 = m_t = 173$ GeV. The left panels show the real parts and the right hand panels the imaginary parts. The upper panels show the approximations produced after using the final TayInt algorithm and by SecDec while the lower panels display the associated errors. This illustrates how TayInt can produce algebraic approximations which evaluate to precise numerical approximations in different kinematic regions even for integrals with highly oscillatory behaviour in those regions.

11.9 Recap

This chapter began with a statement of the problems that arise when the conceptual TayInt algorithm was applied to highly complicated elliptic integrals. These are:

1. The proportion of contour configurations which lead to a representation of the subsector integrands that is suited for a Taylor expansion drops considerably.

2. The subsector integrand surfaces contain many more turning points and each subsector integrand is very different.

3. The subsector integrands vary more as the kinematic scales are changed.

This chapter has described how the final TayInt algorithm makes more potential contour configurations available. This allows an accurate representation of the subsectors of even elliptic integrals beyond leading order in ϵ to be calculated (OT2-5). Once such a

representation has been found, the increased number of turning points in the subsectors is combated by making more potential partition sets available to the algorithm (OT6-9). This allows the precision of the TayInt approximation to be controlled. Finally, the use of kinematic training and cross-validation sets anchors the accuracy of the algebraic approximations irrespectively of the turbulent nature of the subsectors as the kinematic points inserted are altered (OT4, OT5). The final TayInt algorithm is therefore able to produce accurate, precise and general approximations for all of the Feynman integrals calculated using the conceptual algorithm and the more complicated elliptic and non-planar integrals. With this, the final TayInt algorithm is complete. It remains to embed this algorithm in a fully-automated and parallelisable Python program (Chapter 12). A glossary of all the key parts of the final TayInt algorithm is given in Appendix B.1.3.

12 | The TayInt Program

12.1 Introduction

Chapters 9, 10 and 11 describe how the TayInt algorithm was conceptualised, developed and finalised. Since the proof-of-concept, the TayInt method was extended to include two fully robust and layered algorithms for locating the thresholds of a Feynman integral (Chapter 10) and choosing the relevant contours for calculating it in the physical kinematic region, such that the kinematic generality and controlled precision is guaranteed by construction (Chapter 11). Moreover, the dependence of TayInt on the Reduze program has been removed, as the final TayInt algorithm is able to calculate divergent integrals (Chapter 13) and the sector decomposition has been integrated into the TayInt program, using SecDec-3.0.9. internally. The extensions of the TayInt algorithm ensured that it can produce accurate, precise and kinematically generalisable algebraic approximations for Feynman integrals up to the two-loop elliptic level. Evidence for this is provided in Chapter 13, where the results of applying TayInt to elliptic and non-planar integrals are presented.

The motivation for promoting TayInt to a self-contained Python program stems from the mathematical complexity of Feynman integrals and the need to calculate multiple families of integrals in order to perform phenomenological calculations. To prepare TayInt for phenomenological applications, the final algorithm was promoted to a fully-automated, self-contained, parallelisable and fast Python program and this chapter contains a users manual for it. Whilst describing the program, the names of the input fields will be typeset in bold and the name of the directories will be typeset in italics. This is for illustration only.

The workflow of the TayInt program reads as follows:

1. Input an integral.
2. Identify the approximate positions of the thresholds using the TayInt threshold-finding algorithm (step U2 in Table 11.1).
3. Carry out a decomposition into subsectors with smoother integrands. These are obtained using code from SecDec 3 [51–54], which is integrated into TayInt.
4. Find the relevant contours and partition sets to produce a result which can be reliably evaluated at arbitrary kinematic points to generate a precise and accurate

approximation. The over-threshold part of the final TayInt algorithm (steps OT1-9 in Table 11.1) performs this.

5. Perform the integration to generate the library of systematic approximations (step OT10 in Table 11.1).

The TayInt program is based on a divide and rule philosophy in order to produce an algebraic result in the kinematic scales for a generic Feynman integral G without performing the integral analytically. The advantage of this is that no mathematical understanding of the integral is needed. The disadvantage is that the program must be able to bring any Feynman integrand into a form with a well-converging Taylor expansion. To do this TayInt breaks down the calculation. As such, the Feynman integral is computed in full at orders $\epsilon^0, \epsilon^1, \epsilon^2$ via conformal mappings for kinematic points in the Euclidean or below-threshold Minkowski regions. This was discussed fully in Chapter 9. For those kinematic points which are over thresholds of the integral, TayInt chooses a contour and partition set for each subsector G_l of the Feynman integral in each threshold region that allows an accurate algebraic approximation to be calculated. These approximations are then combined to yield a full result for the Feynman integral that is valid in over-threshold regions.

12.2 The Structure of the TayInt Program

The file structure of the TayInt program is shown in Fig. 12.1. The key constituents are described below.

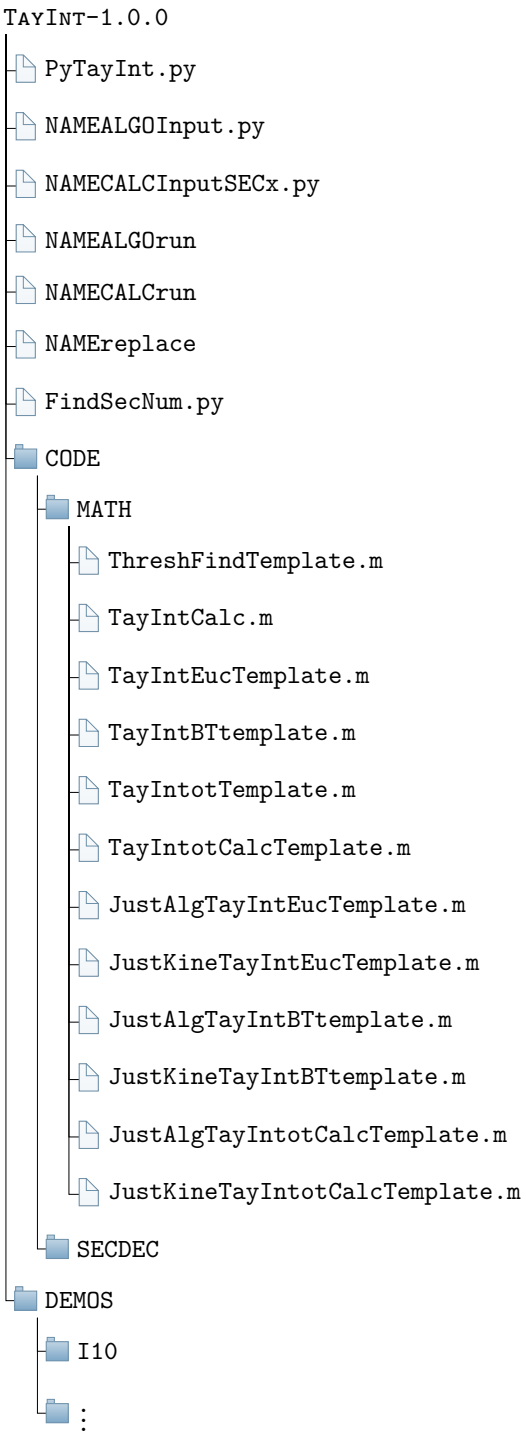


Figure 12.1: The structure of the TayInt program.

12.2.1 Python Program

PyTayInt.py is the Python program that runs the final TayInt algorithm. It contains breaks to inform the user of the important milestones as the algorithm proceeds, such as positions of the thresholds and the chosen contours and any numerical approximations. The program contains two parts. The first is the final TayInt algorithm, which is run once. The second is the calculation of the Feynman integral using the chosen contours and partition sets determined using that algorithm. This is done for each subsector of the Feynman integral. To initiate these, the user provides the necessary input information by editing the template files NAMEALGOInput.py and NAMECALCInputSECx.py, changing NAME to that of the integral under consideration and copying it to a new directory in which this integral will be calculated.

12.2.2 The Input Files

In the DEMOS subdirectory, examples of the application of the TayInt program to various integrals are provided. In the case of the I10 integral, the input file I10ALGOInput.py reads:

```
# GENERAL FIELDS: THE INFORMATION THE USER MUST PASS TO THE ALGORITHM

HOMdir = "PathToTayInt/TayInt-1.0.0"

WORKdir1 = "PathToTayInt/TayInt-1.0.0/DEMOS/Results"

NAMEvar1 = "I10"

WORKname = "I10demo"

EPSvar1 = "{0,1,2}"

KINEvar1 = "{q23s,q123s}"

KINEvar2 = "{m,m,m,0}"

KINEvar3 = "{0.5*29929}"

KINEvar4 = "{173}"

KINEvar5 = "{4*29929}"

KINEvar6 = "{36*29929}"

KINEvar7 = "25"

LOOPvar1 = "2"

PROPvar1 = "{(l+q123)^2-ms,(l+q23)^2-ms,}"

POWvar1= "{1,1,2,1}"
```

```
EXTMOvar1 = "{q123,q1,q23}"

LEGSvar1 = "3"

SPRvar1= "{SP[q1,q1]→ 0,SP[q23,q23]→q23s,SP[q123,q123] →q123s,
SP[q23,q123]→0.5 · (q123s+q23s)}"

SECvar2 = "1"

TAYaltord1 = "6"

TAYaltstart1 = "2"

# CONDITIONAL FIELDS: THIS SET UP RUNS THE FULL FINAL TAYINT
ALGORITHM AND SHOULD ONLY BE CHANGED IF THE USER
WANTS TO RUN A SPECIFIC PART AND UNDERSTANDS HOW THE ALGORITHM WORKS

SECDECchoice=y

THRESHchoice=y

ALGOchoice=y

CALCchoice=n

ALGchoice=n

NUMchoice=n

TAYchoice=n

TAYstartchoice=n ,
```

The quotation marks also feature in the actual input file and are there to ensure that the users input is converted to a Python string. The user input should always be enclosed within quotation marks. Once the quotation marks close, the input ends. The input file `I10CALCInputISECx.py` has the same layout as its algorithmic counterpart and the fields which take different entries in the full TayInt setup are given below:

```
# GENERAL FIELDS: THE INFORMATION THE USER MUST PASS TO THE ALGORITHM

SECvar2 = "SECVAR"

# CONDITIONAL FIELDS: THIS SET UP RUNS THE FULL FINAL TAYINT
ALGORITHM AND SHOULD ONLY BE CHANGED IF THE USER
WANTS TO RUN A SPECIFIC PART AND UNDERSTANDS HOW THE ALGORITHM WORKS

SECDECchoice=n

THRESHchoice=n
```

```
ALGOchoice=n

CALCchoice=y

ALGchoice=y

NUMchoice=y

TAYchoice=n

TAYstartchoice=n ,
```

which will run the full (algebraic and numerical) calculation for each subsector (the number of subsectors is automatically obtained by the launch file `NAMEALGOrun` after the sector decomposition has been carried out and passed to `NAMECALCrun`.

Descriptions of each general input field are now provided. Note that Mathematica syntax must be used for the input, because it will be inserted into Mathematica `.m` files.

- `HOMdir`: This is the path to the TayInt folder.
- `WORKdir1`: this is where the working directory for this calculation will be generated.
- `NAMEvar1`: This is the name of the integral being calculated. For compatibility it is best to use the name created when the integral family was generated. It will subsequently appear in all the files that are created by TayInt.
- `WORKname`: The name of the folder created in `WORKdir1`.
- `EPSvar1`: This is a list of the epsilon orders at which the integral will be calculated.
- `KINEvar1`: The list of kinematic scales in the integral, each written as compact symbols, with the scale to be varied in the first (leftmost) position, written in the format `"{q23s,q123s}"` *not* `"{(q2+q3)**2,(q1+q2+q3)**2}"`.
- `KINEvar2`: The list of masses for each propagator, including zero masses and identical masses, written as `"{m,m,m,0}"` *not* `"{m}"`.
- `KINEvar3`: The list of constant values for the scales in `KINEvar1` that the user wants to fix. This can be written as an arithmetic operation, or as a floating point or integer number but always in Mathematica syntax (as these numbers will be inserted into the `TayIntotAlgoTemplate.m` file).
- `KINEvar4`: The list of distinct values for the masses in `KINEvar2`.
- `KINEvar5`: The minimum value for the scale in `KINEvar1` that will be varied, which can be written in arithmetic form or as a number, always in Mathematica syntax (as these numbers will also be inserted into the `TayIntotAlgoTemplate.m` file).

- `KINEvar6`: the maximum value for the scale in `KINEvar1` that will be varied, which can be written in arithmetic form or as a number, but always in Mathematica syntax (once again, these numbers will be inserted into the `TayIntotAlgoTemplate.m` file).

- `KINEvar7`: The number of points in the range [`KINEvar5`,`KINEvar6`] at which numerical approximations for the integral will be produced. If only the algebraic approximations are required, there is a subsequent option to disable generating numerical approximations, however numbers for `KINEvar5`-`KINEvar7` should still be entered regardless.

- `LOOPvar1`: The number of loops in the integral.

- `PROPvar1`: The list of propagators in the form `"{(l+q123)^2-ms,(l+q23)^2-ms,}"`, where `k` and `l` are the loop momenta and `ms` is the squared mass. The loop momenta *must* be taken from the list `{k,l,m,n,o,p,q,r,s,t}`. They *must* not be entered as `k1,k2,k3,...`. The masses entered here should be the exact symbols that appear at the corresponding position in `KINEvar2` with an `s` appended.

- `POWvar1`: the powers of the propagators in `PROPvar1` must be entered in the corresponding positions in this list, such as `"{1,1,2,1}"`, which means the third propagator in the list `PROPvar1` is squared.

- `EXTMOvar1`: The list of the external momenta.

- `LEGSvar1`: the number of external legs the diagrammatic representation of the Feynman integral has. This is an integer.

- `SPRvar1`: The list of scalar product rules which define the kinematic properties of the integral.

- `SECvar2`: An integer number denoting which subsector of the Feynman integral is to be computed. This should *not be edited by the user* and must remain as entered in the template files, `"1"` in that of the algorithm and `"SECVAR"` in that which is used for the calculation. It will be replaced automatically by the launch file as explained later. The division into subsectors allows the calculation to be parallelised.

- `TAYaltord1`: The integer that denotes the order to which the Taylor expansion of the integrand will be computed if the user chooses not to use the default value of `4`. Changing the order is not advised unless the integral to be calculated is especially straightforward.

- `TAYaltstart1`: The integer that denotes the starting order from which the Taylor expansion will be computed if the user chooses not to use the default value of `0`. This may be the case if several orders were already computed and means that the user does not need to waste time recomputing those orders of the expansion.

As TayInt contains several tools for studying Feynman integrals, it is important that they can all be used in isolation or in custom groups depending on the needs of the user, rather than just as one large inflexible calculation. The user can control this by entering `y` or `n` for the following conditional input fields:

- **`SECDECchoice`**: Either `y` or `n`. This controls whether or not TayInt uses SecDec to perform the sector decomposition. The integral will be decomposed into all of its subsectors, so this only needs to be done once per ϵ order. In all cases therefore, this is `y` in the input file corresponding to the algorithm and `n` in that of the calculation.

- **`THRESHchoice`**: Either `y` or `n`. If `y` is entered, TayInt will compute the locations of the thresholds of the integral, the algorithm for which was described in Chapter 10. This only needs to be done once per integral, so this should be `y` in the input file of the algorithm and `n` in that of the calculation.

- **`ALGOchoice`**: Either `y` or `n`. If `y` is chosen then the over-threshold part of the algorithm, described in its final form in Chapter 11 is run to determine the contour configuration needed to produce accurate, precise and kinematically generalisable approximations for each subsector in the over-threshold regions. This only needs to be performed once per integral. In all cases therefore, this is `y` in the input file corresponding to the algorithm and `n` in that of the calculation, unless the algorithm has already been run, in which case it should be `n` everywhere.

- **`CALCchoice`**: Either `y` or `n`. If `y` is entered, then the calculation part of the code is run, according to the choices for `ALGchoice` and `NUMchoice` as described below. If `n` is entered, then no calculation, algebraic or numerical, will be performed, as is the case in the algorithm's input file.

- **`ALGchoice`**: Either `y` or `n`. If `y` is entered, then the algebraic computation of the integral will be performed. This only needs to be done once per subsector, so, unless the algebraic approximations are already present, this should be `y` in the calculation's input file and `n` in the algorithm's input file.

- **`NUMchoice`**: Either `y` or `n`. If `y` is entered, then the numerical calculation of the integral will be performed. This is only possible if `y` was entered for `ALGchoice` or if the algebraic calculation of the subsector in question has already been performed. This is always `n` in the input file of the algorithm.

- **`TAYchoice`**: Either `y` or `n`. If `y`, the ending order of the Taylor expansion will be changed to **`TAYaltord1`**. If `n`, the ending order of the Taylor expansion will remain as the default value of `4`. It is advised to always use `n` here.

- **`TAYstartchoice`**: Either `y` or `n`. If `y`, the starting order of the Taylor expansion will be changed to **`TAYaltstart1`**. If `n`, the starting order of the Taylor expansion will remain as the default value of `0`.

The user is advised to only start altering this conditional entries once familiar with the TayInt program.

12.2.3 Launch Files

As shown in Fig. 12.1, the TayInt Python program is not the only file immediately within the TayInt-1.0.0 directory. It also contains the TayInt launch files, which trigger the launch of the algorithm and the calculation respectively, as follows:

- `NAMEreplace`: This simple script inserts the name for the integral to be calculated into the `NAMEALGOrun` and `NAMECALCrun` files. The user must simply replace `NAMEvar1` by their entry for this field in the input files and then run the script.
- `NAMEALGOrun`: This script inserts the input provided in `NAMEALGOinput.py` into the Python program `PyTayInt.py`, thus generating the file `FinPyTayIntNAMEALGO.py` which runs the final TayInt algorithm. It does this by replacing symbols in the relevant template files in the *`CODE/MATH`* subdirectory by the values entered by the user in the input file, for example $K6 \rightarrow$ KINEvar6. These template symbols are `$A1$,$B1$,$C1$,$E1$,$TD1$,$D1$-$D10$,$K1$-$K7$, $S1$-$S2$,$T1$-$T2$`. The user must **never** use any of these symbols, or any of the input fields in `NAMEInputSECx.py`, as names of an integral or the working directory as this will lead to a doubly defined name. After the final TayInt algorithm has finished, the `NAMEALGOrun` script calls the Python code `FindSecNum.py` to determine the number of subsectors that the Feynman integral has at the ϵ order entered in the input file and inserts it into `NAMECALCrun`, generating the file `NAMECALClaunch`.
- `NAMECALClaunch`: This script launches the TayInt calculation of the Feynman integral. It does not need to be altered in any way and can simply be run by the user to enact the calculation of the integral along the lines previously specified in `NAMECALCInputSECx.py`. If the user wishes to parallelise the calculation of the subsectors, then the symbol Y must be replace by the desired number of concurrent subprocesses.

12.2.4 Code Files

Once the TayInt algorithm or calcuation has been launched, the Python program calls on the necessary Mathematica code from the *`CODE`* subdirectory. These are:

Functions

- `TayIntCalc.m`: the functions that perform the actual Taylor expansion and integration, which are described in detail in Appendix B.1.
- `TayIntfunctions.m`: the generic functions that are needed for input and output of results stored by the algorithm.

Threshold Location

- `ThreshFindTemplate.m`: this is the Mathematica file that performs the threshold-finding part of the algorithm, step U2 in Table 11.1.

Euclidean and Below-Threshold Calculation

- `TayIntEucTemplate.m`: Performs the calculation of the Feynman integral (which requires no algorithm) for Euclidean kinematics.
- `TayIntBTtemplate.m`: Performs the calculation of the Feynman integral for below-threshold Minkowski kinematics (step BT2).
- `JustKineTayIntEucTemplateMac.m`: Allows the user to only perform the numerical part of the calculation for Euclidean kinematics.
- `JustKineTayIntBTtemplateMac.m`: Allows the user to only perform the numerical part of the calculation for below-threshold Minkowski kinematics.

Over-Threshold Calculation

- `TayIntotTemplate.m`: This performs the over-threshold part of the algorithm that allows the subsectors of the Feynman integral to be calculated as algebraic functions which will give accurate, precise approximations for arbitrary over-threshold Minkowski kinematics.
- `TayIntotCalcTemplate.m`: This file calculates algebraic and numerical approximations for each subsector of the Feynman integral and then combines them to generate the full result.
- `JustAlgTayIntCalcTemplate.m`: This allows the user to perform only the algebraic part of the over-threshold calculation.
- `JustKineTayIntCalcTemplate.m`: This allows the user to perform only the kinematic part of the over-threshold calculation.

12.3 The Demonstration of the Program

The TayInt program consists of multiple stages so several results must be stored in the directory given by the user as `WORKdir1`. For example, the results of the final TayInt algorithm are stored in a subdirectory of `WORKdir1` known as `OTsetup`. In Fig. 12.2 the structure of `WORKdir1` (found in *`TayInt-1.0.0/DEMOS/`*) for the case of the integral I10 (which has eight subsectors) is depicted. This illustrates the locations of the Mathematica files containing the final algorithm and calculation code. In the subsequent Figs. 12.3-12.8, the structure of the subdirectories in which results are stored is depicted. Only the files used for computing results up to order ϵ^0 are displayed, for brevity.

12.3.1 Directory Structure Diagrams

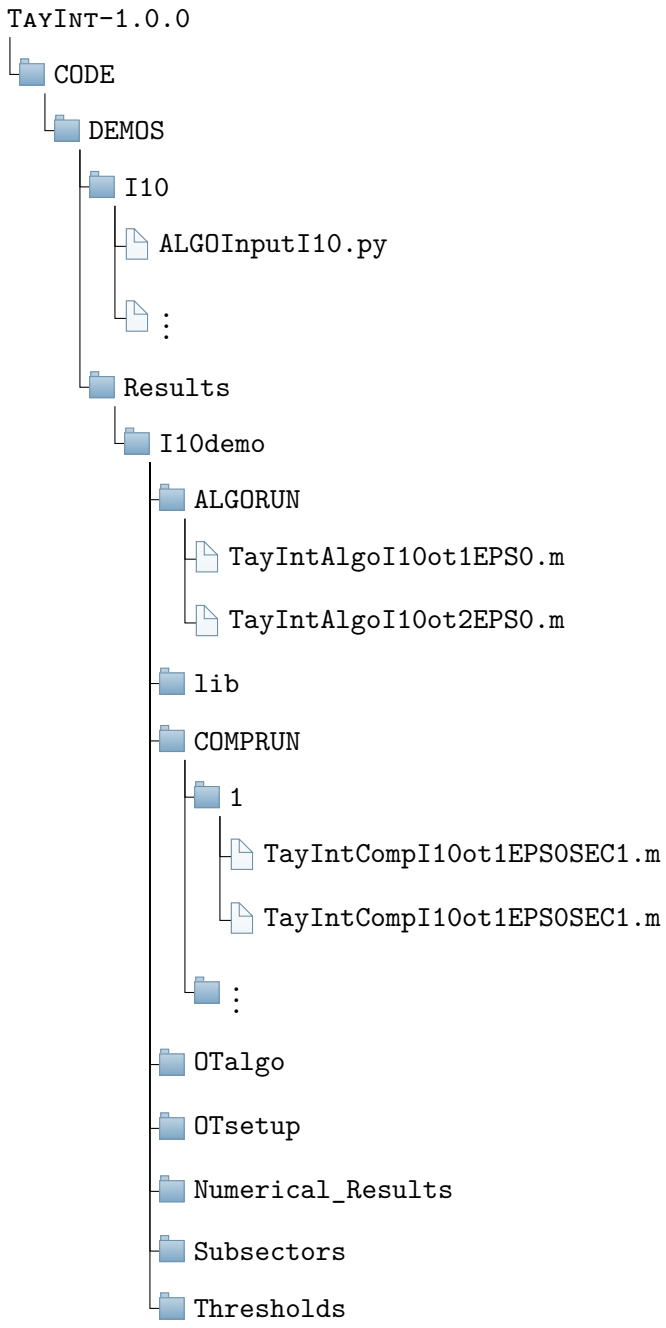


Figure 12.2: The structure of the working directory of the TayInt program in the case of the integral I10, found in the *DEMOS* subdirectory of the TayInt program.

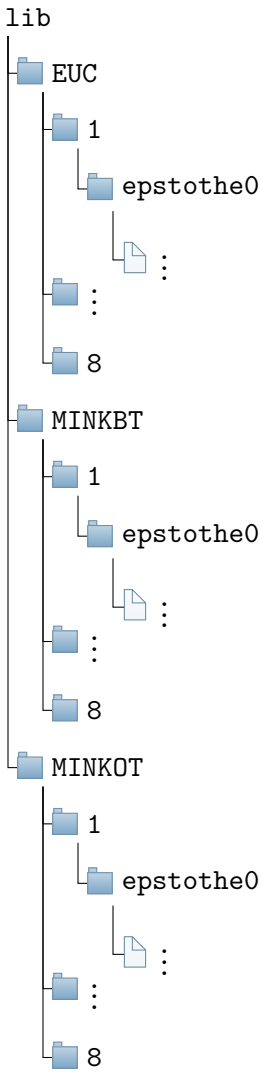


Figure 12.3: The structure of the *lib* subdirectory of the working directory of the TayInt program.

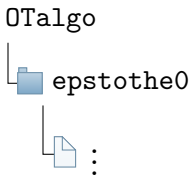


Figure 12.4: The structure of the *OTalgo* subdirectory of the working directory of the TayInt program.

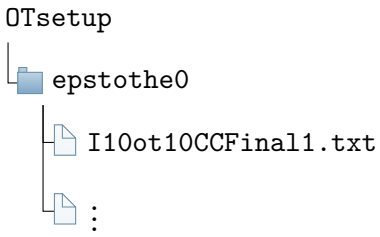


Figure 12.5: The structure of the *OTsetup* subdirectory of the working directory of the TayInt program, which contains the output of the final algorithm, for example `I10otCCFinal1.txt`, the chosen contour for subsector `1` of I10 in the first over-threshold region.

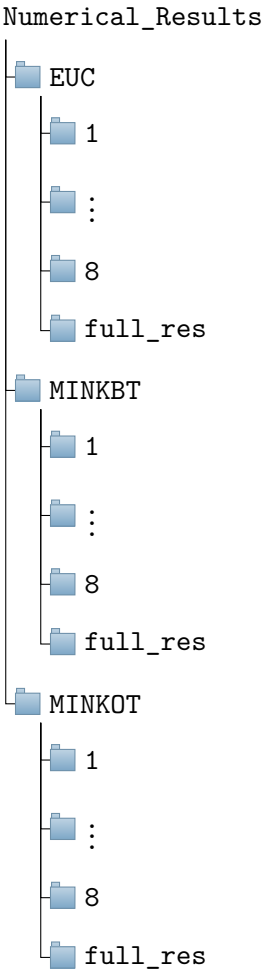


Figure 12.6: The structure of the *Numerical_Results* subdirectory of the working directory of the TayInt program.

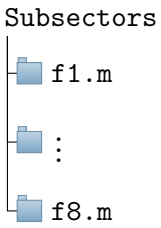


Figure 12.7: The structure of the *Subsectors* subdirectory of the working directory of the TayInt program.

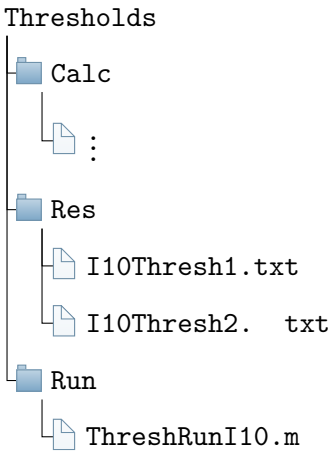


Figure 12.8: The structure of the *Thresholds* subdirectory of the working directory of the TayInt program.

12.3.2 Directory Structure Descriptions

The subdirectories within `WORKdir1`, which is called `I10demo` in the case of the I10 integral, are as follows:

1. `Subsectors`: Contains the subsectors of the Feynman integral, sorted into subdirectories for each epsilon order. In the case of I10, there are eight for each order, `f1.m`,...`f8.m`.

2. `Thresholds`: Contains the subdirectories `Calc`, `Res` and `Run`. In `Run`, the Mathematica file which implements the threshold-finding part of the final TayInt algorithm (step U2 in Table 11.1) is stored, termed `ThreshFindI10.m`. In `Calc`, the algebraic approximations produced during step U2 (Table 11.1) of the final TayInt algorithm are stored. They do not need to be sorted into subdirectories for each kinematic region or ϵ order as the thresholds are a property of the integral as a whole and inform the program which kinematic regions to use. In `Res`, the located thresholds are stored, to be used in determining how many regions the

over-threshold part of the final TayInt algorithm needs to be run in. These are `I10Thresh1.txt` and `I10Thresh2.txt` in the case of I10.

3. `ALGORUN`: Contains the Mathematica files which implement the over-threshold part of the final TayInt algorithm (steps OT1-9 in Table 11.1) in each over-threshold region detected by the threshold-finding algorithm (step U2 in Table 11.1).

4. `COMPRUN`: This contains the Mathematica files which implement the calculation part of the final TayInt algorithm (steps BT2 and OT10 in Table 11.1) in the below-threshold region and in each over-threshold region detected by the threshold-finding algorithm (step U2 in Table 11.1).

5. `OTalgo`: Contains the algebraic approximations used by the over-threshold part of the final TayInt algorithm to determine the contour configurations and partition sets for each subsector, which are stored in

6. `OTsetup`: All the results of the over-threshold part of the final TayInt algorithm which are needed to produce accurate, precise and kinematically generalisable algebraic approximations for the Feynman integral, which are stored in,

7. `lib`: The library of algebraic approximations for the Feynman integral, sorted into further directories denoting the pertinent kinematic region `EUC`,`MINKBT`, `MINKOT`, subsector, `1`,...,`8` in the case of I10 and ϵ order.

8. `Numerical_Results`: The numerical approximations for the Feynman integral at the kinematic points entered by the user in **KINEvar5**-**KINEvar7**, sorted into further directories denoting the pertinent kinematic region `EUC,MINKBT,MINKOT`, subsector, `1`,...,`8` in the case of I10 and ϵ order.

The flow of information between the TayInt home directory and the relevant working directory is illustrated in Fig. 12.9. The smudged, solid and dashed lines denote directories in the home directory, in the working directory and files, respectively. Red denotes ϵ^0, blue denotes ϵ^1, green denotes ϵ^2. It is important to stress that TayInt produces algebraic approximations for the entire Feynman integral that are valid in the Euclidean or below-threshold Minkowski regions and sector by sector algebraic approximations that are valid in the over-threshold Minkowski regions. These sector by sector approximations are found in the *calc/MINKOT/i/epstothej* directory, where i runs over the total number of sectors and j over the total number of epsilon orders. If calculated, the numerical approximations in the kinematic region specified by the user are moved to the directory *Numerical_Results/MINKOT/i/epstothej* and then summed over all the sectors to give a result for the full Feynman integral. This is moved to the directory *Numerical_Results/MINKOT/full_res/epstothej*. The details of the workflow are now summarised:

1. **Sector Decomposition**: The Feynman integral is reduced to a number of subsector integrals (step U1 in Table 11.1). The `SECDEC` directory contains the template files needed to perform the decomposition with SecDec-3.0.9 which are

edited by the Python program using the input from the user to generate specific SecDec run files containing the name of the integral being calculated. SecDec-3.0.9 is then run internally using the algebraic option to quickly decompose the Feynman integral into its subsector integrals, which are then stored in the *Sectors/epstothej* directory and are called as input into the `TayIntAlgo*.m` files.

2. **Threshold Locations**: The locations of the thresholds (step U2 in Table 11.1) in the scale the user wants to vary (the first one in **KINEvar1**) are determined. The *CODE* directory contains the Mathematica template `ThreshFindTemplate.m` which is edited by the python script according to the input of the user. This file is stored in the **Run** subdirectory of the *Thresholds* directory, for example, in the case of `I10demo`, this is called `ThreshRunI10.m`. The positions of the thresholds are stored in the **Res** subdirectory of the **Thresholds** directory, in the case of `I10demo` they read `I10Thresh1.txt` and `I10Thresh2.txt`. These files are then accessed by the `TayIntAlgo*.m` files and used when determining the contours to avoid the threshold discontinuities in the over-threshold regions (steps OT1-9 in Table 11.1).

3. **Contour Choice**: The most significant step in calculating integrals with TayInt is determining the contour of integration which allows accurate, precise and kinematically generalisable approximations to be produced for each subsector of a generic Feynman integral which are valid in the over-threshold kinematic regions. The full list of possible contours is generated and the `TayIntAlgo*.m` files choose which one produces an integrand that is best suited to achieve objectives O1-3 of the final TayInt algorithm via a Taylor expansion in the Feynman parameters. For an integral with three subsectors and eight possible contours, the list of contour choices would be a list of three integers between 1 and 8, such as $\{1, 4, 7\}$. A list of contour choices is calculated for every distinct over-threshold region encompassed by the users kinematic inputs via **KINEvar5-7**. The `TayIntAlgo*.m` files are named according to which integral, region, epsilon order they are computing contours for. The general form of the name is `TayIntAlgo[IntegralName][Threshold Region][EpsilonOrder].m`. So, in Fig. 12.8 the names of the **ALGORUN** files all contain `ot1` as the kinematic identifier. This is because the user requested approximations in the region between the first threshold, at $s = 4m_t^2 = 119716$ GeV^2 and the second threshold, $s = 9m_t^2 = 269361$ GeV^2. If the user had entered values that also crossed into the second threshold region, then there would be three more **ALGORUN** files with names starting with `TayIntAlgoI10ot2`.

4. **Algebraic and numerical approximations**: TayInt produces an algebraic approximation for each subsector in each distinct over-threshold region of the Feynman integral by using the contours that were found using the `TayIntAlgo*.m` files. These approximations are contained in the *calc/MINKOT* subdirectory of the TayInt working directory, for example *I10demo*. The approximations for the different threshold regions are all contained in the same folder and are used to produce a complete list of numerical approximations encompassing the entire kinematic re-

gion that the user requested via **KINEVAR5-7**. Euclidean and below-threshold Minkowski approximations would be contained in the *calc/EUC* and *calc/MINKBT* subdirectories, respectively. The numerical approximations are transferred to the *Numerical_Results* subdirectory and are further sorted into the kinematic region, the subsector number (if MINKOT) and the epsilon order. For the over-threshold calculation, the numerical approximations for each sector are added together and stored in the *calc/MINKOT/full_res* subdirectory according to epsilon order.

Now that the automated TayInt program has been described, its capability to produce accurate, precise and kinematically generalisable algebraic approximations will be displayed in the following chapter, for Feynman integrals up to the complexity of two-loop, four-point elliptic integrals.

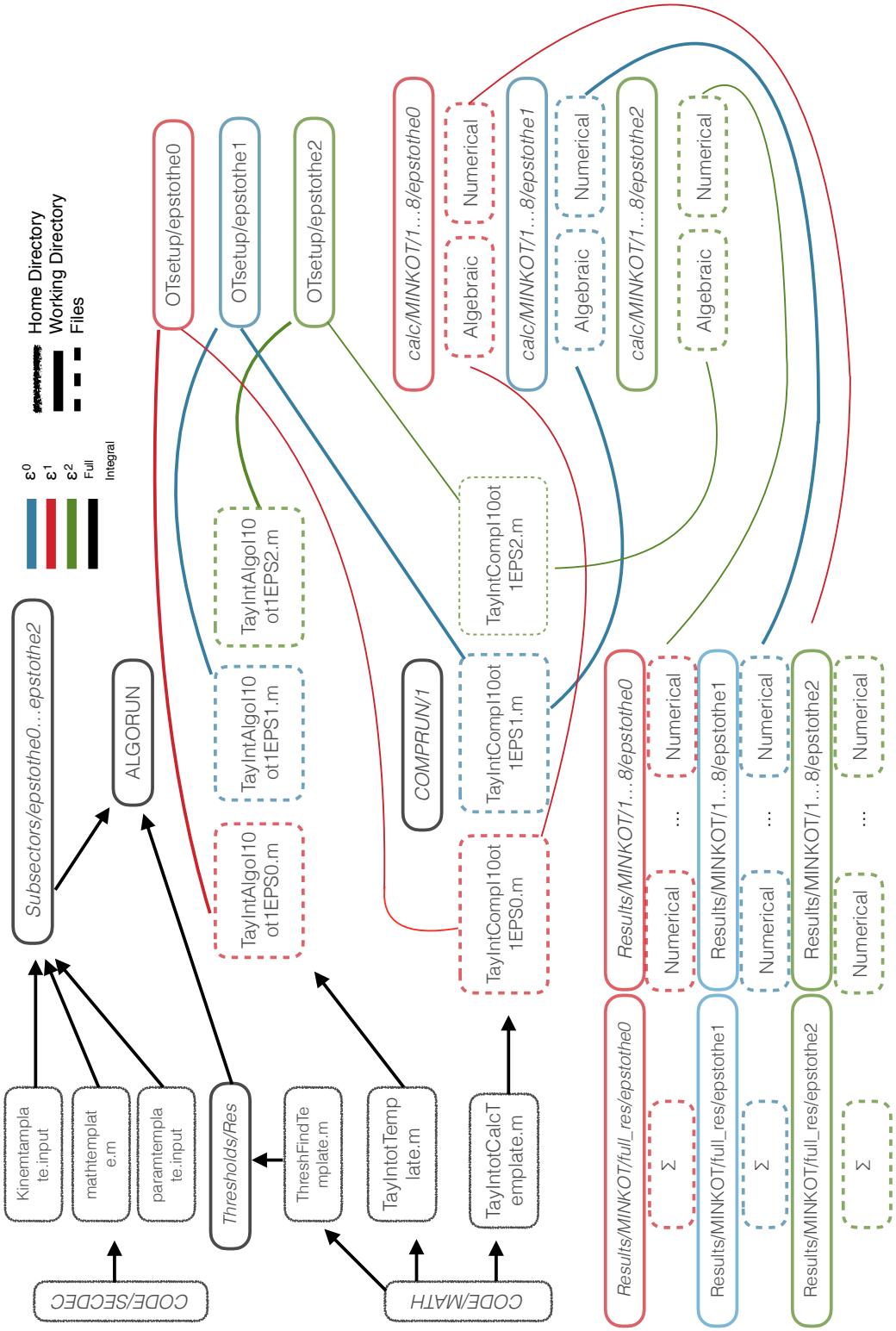


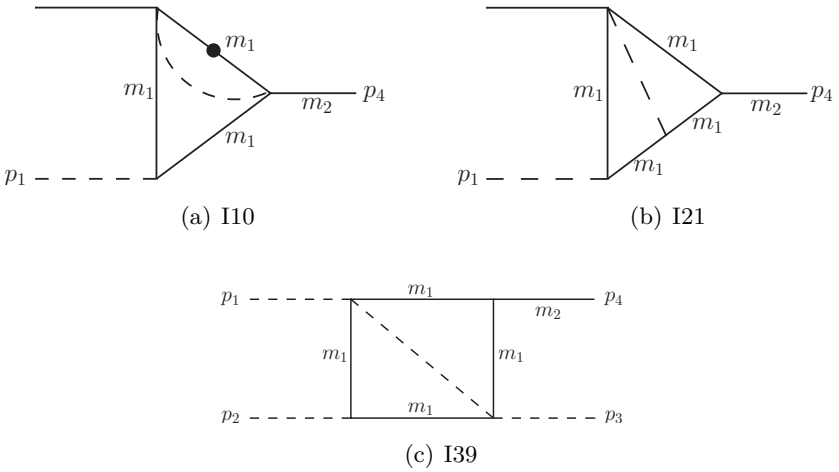
Figure 12.9: The flow of information between the TayInt home directory and the working directory, depicted in the case of I10. The smudged lines denote the folders of the home directory, the solid lines those of the working directory and the dashed lines files within them. The black lines represent quantities or files pertinent to the entire Feynman integral and blue, red and green denote the epsilon orders ϵ^0, ϵ^1 and ϵ^2 respectively.

13 | Results and Discussion

13.1 Introduction

It now remains to demonstrate that the TayInt program can indeed achieve the objectives O1-3 set out in Chapter 11 and produce accurate systematic approximations for two-loop, four-point integrals up to the elliptic level. The numerical accuracy and precision of these approximations improves by means of partitioning the integral, for arbitrary kinematic values of the scales.

13.2 Application to Three- and Four-Scale, Two-Loop Four-Point Integrals



(a) I10 (b) I21

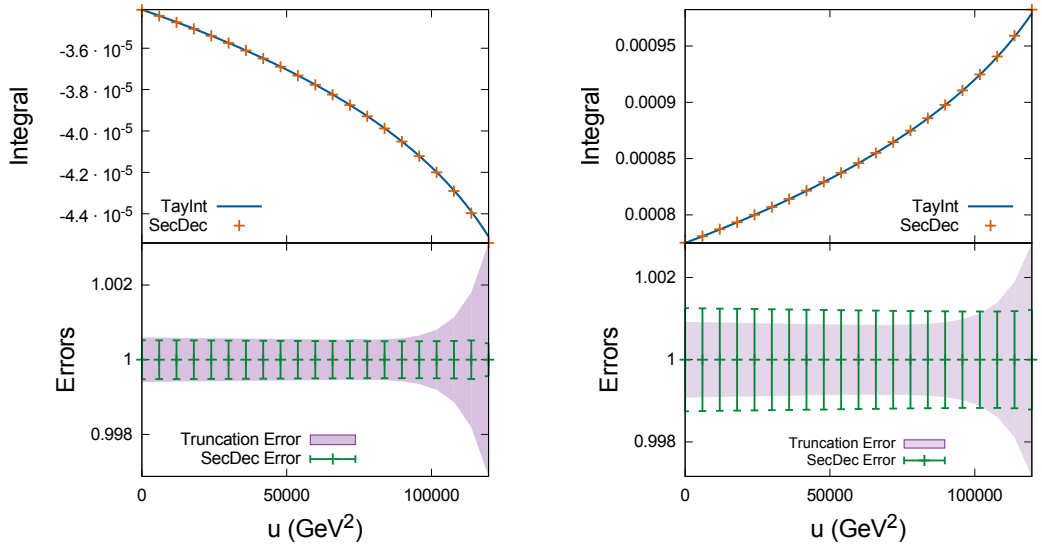
(c) I39

Figure 13.1: The triangle I10 ((a)) and I21 ((b)) graphs for which analytical results are available [92]. The box-type integral I39, (c), thus far unknown analytically. Dashed lines indicate massless, solid internal lines massive, and dots squared propagators. Solid external lines denote, where indicated, massive and else off-shell particles.

To begin with, algebraic approximations for the three- and four-scale, two-loop, four-point integrals I10, I21 and I39, shown in Fig. 13.1 are presented below and over threshold and for different orders in ϵ, respectively. Below threshold, four orders and three partitions are universally used for the integrated Taylor expansion. Over threshold, four orders are used for the Taylor expansion and the number of partitions per subsector is determined by the final TayInt algorithm.

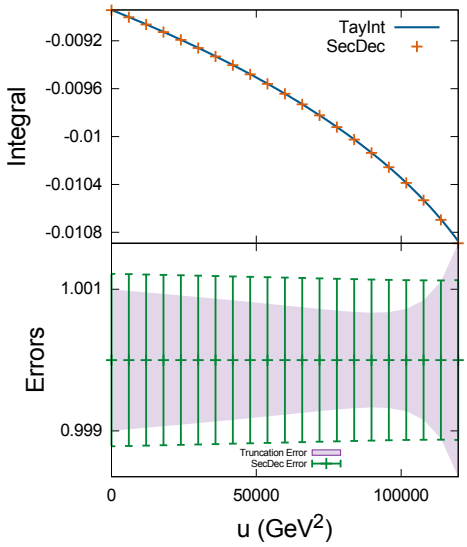
In Fig. 13.2, approximations for the finite I10 integral below threshold and up to $\mathcal{O}(\epsilon^2)$ are shown. In the upper half of the plots, the numerical evaluation of the TayInt algebraic approximation is shown as a blue solid line, overlaid with results generated with the program SecDec, depicted as orange crosses. The only deviation (though hardly noticeable) can be seen directly on and around the threshold at $u = 4\,m_1^2 = 4m_t^2 = 119716\ \text{GeV}^2$, where the difference between the TayInt approximation and the SecDec result reaches at most 0.7%. In the lower half of the plots, the uncertainty band of TayInt is shown in lilac and can be compared to the uncertainties coming from SecDec shown in green. The SecDec results were computed using default numerical integration parameters and the integrator Vegas [131], asking for a relative accuracy of 10^{-3}. The relative accuracy is adapted to the accuracy of the TayInt approximations. The same colour coding is used for all subsequent plots in this chapter.

There is no appreciable precision loss as the order in ϵ increases. For I10 the mean $\frac{\text{SecDec}}{\text{TayInt}}$ ratios are $1.0006, 1.00039, 1.00021$ at ϵ^0, ϵ^1 and ϵ^2, respectively.



(a)

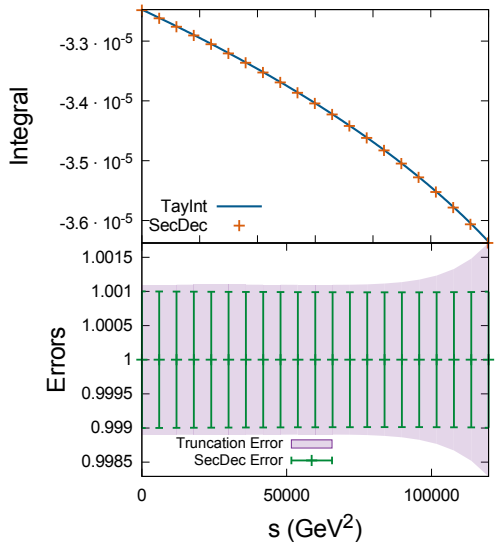
(b)



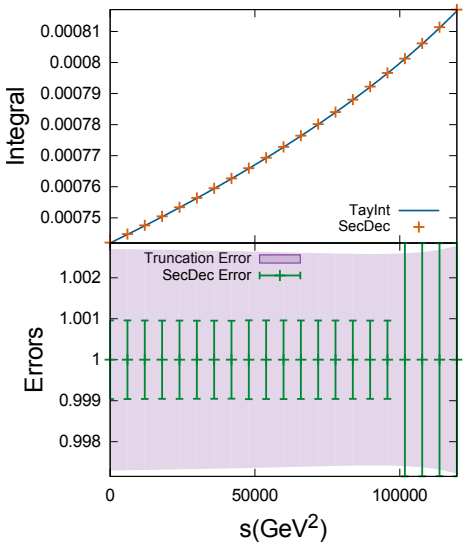
(c)

Figure 13.2: I10 below threshold, calculated with four orders and three partitions at; (a) $\mathcal{O}(\epsilon^0)$, (b) $\mathcal{O}(\epsilon^1)$, (c) $\mathcal{O}(\epsilon^2)$ with $m_2 = \frac{1}{\sqrt{2}} m_1$ and $m_1 = 173$ GeV. The lower plots show the relative uncertainties of the TayInt approximations and numerical SecDec results, respectively.

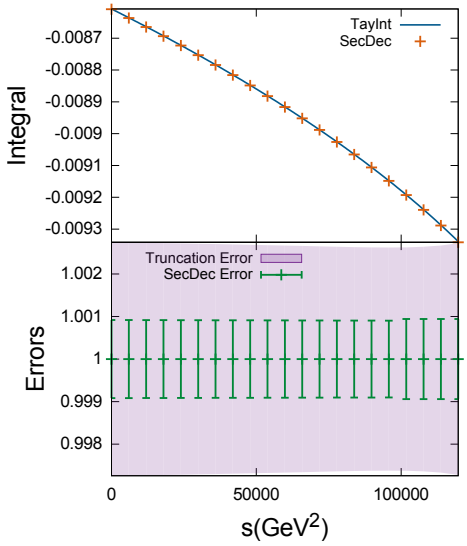
In Fig. 13.3, approximations for the finite I39 integral up to $\mathcal{O}(\epsilon^2)$ are plotted for s below threshold, asking for a relative SecDec accuracy of 10^{-3}. Again, the maximal deviation between the TayInt approximations and SecDec results can be found on the threshold of $s = 4\,m_1^2$ at $\mathcal{O}(\epsilon^0)$. Here the difference reaches 0.1%, rounded up. There is no appreciable precision loss as the order in ϵ increases. In fact, quite the contrary, the mean $\frac{\text{SecDec}}{\text{TayInt}}$ ratios are $1.0003, 1.00017, 1.00009$ at $\epsilon^0, \epsilon^1, \epsilon^2$ respectively.



(a)



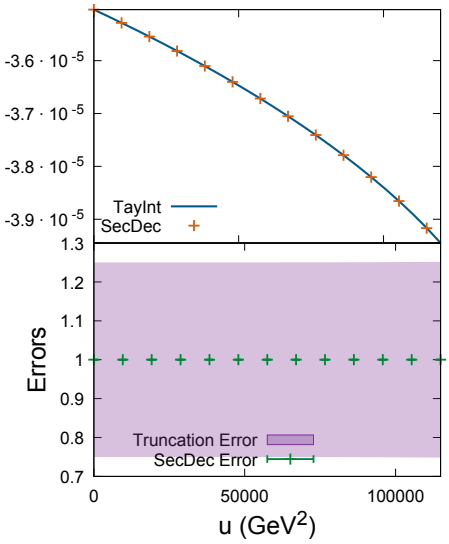
(b)



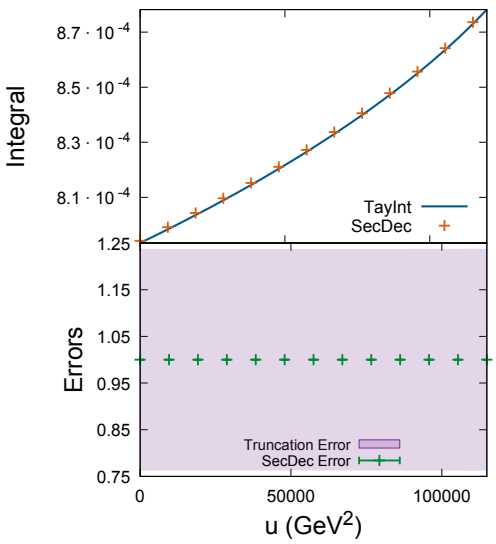
(c)

Figure 13.3: I39 below threshold, calculated with four orders and three partitions at; (a) $\mathcal{O}(\epsilon^0)$, (b) $\mathcal{O}(\epsilon^1)$, (c) $\mathcal{O}(\epsilon^2)$ with $u = -59858$ GeV^2, $m_2 = \frac{1}{\sqrt{2}}m_1$ and $m_1 = 173$ GeV. The lower plots show the relative uncertainties of the TayInt approximations and numerical SecDec results, respectively.

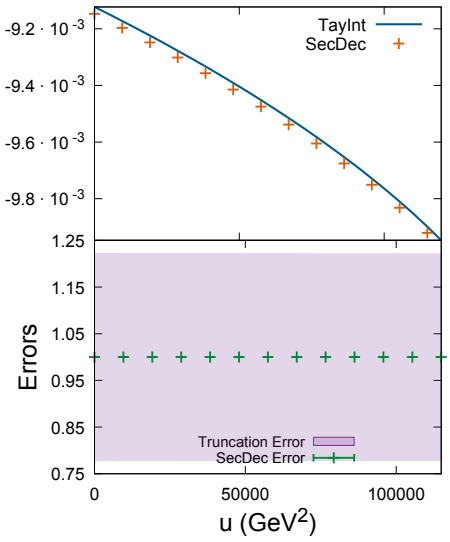
In Fig. 13.4, approximations for the finite I21 integral up to $\mathcal{O}(\epsilon^2)$ are plotted for u below threshold, asking for a relative SecDec accuracy of 10^{-3}. Unlike I10 and I39, the maximal deviation between the TayInt approximations and SecDec results can be found on the threshold of $u = 4\,m_1^2$ at $\mathcal{O}(\epsilon^2)$, where the difference reaches 0.2%, rounded up. There is no appreciable precision loss as the order in ϵ increases, although, unlike I10 and I39, the precision does decrease slightly with ϵ: the mean $\frac{\text{SecDec}}{\text{TayInt}}$ ratios are $1.00001, 1.00084, 1.00246$ at $\epsilon^0, \epsilon^1, \epsilon^2$ respectively.



(a)



(b)



(c)

Figure 13.4: I21 below threshold, calculated with four orders and three partitions at; (a) $\mathcal{O}(\epsilon^0)$, (b) $\mathcal{O}(\epsilon^1)$, (c) $\mathcal{O}(\epsilon^2)$ with $m_2 = \frac{1}{\sqrt{2}} m_1$ and $m_1 = 173$ GeV. The lower plots show the relative uncertainties of the TayInt approximations and numerical SecDec results, respectively.

In Fig. 13.5 the TayInt approximation for I10 at $\mathcal{O}(\epsilon^0)$, obtained with a fourth-order Taylor expansion and a high-uniform partition set for each subsector, is plotted and compared to results from the program SecDec. The SecDec results were computed using default numerical integration parameters and the integrator Vegas, asking for a relative accuracy of 10^{-4}. The plot shows the dependence on the scale u in the threshold region around $u = 4\,m_1^2 \sim 120000$ GeV^2 and above the threshold. By referring to Fig. 13.2(a), a smooth transition from the below-threshold expansion to the over-threshold expansion can be observed, demonstrating the achievement of objective O3 of the final TayInt algorithm.

The size of the error directly on threshold is rooted in the fact that it displays a Landau singularity, a physical discontinuity, for which a Taylor series expansion has to break down by construction. Nonetheless, even on the threshold, the relative accuracy of the TayInt approximation only drops to 10^{-2}. To generate the SecDec results for Figs. 13.5 and 13.6 a relative accuracy of 10^{-4} and the integrator Vegas were chosen. Figure 13.6 is a zoom into the region of larger u values. The TayInt truncation errors in the lower half of the plot are shown in yellow, using a four-fold uniform partitioning and in lilac, using an eight-fold uniform partitioning. A strong increase in accuracy can be observed when the integrand is partitioned more often. More specifically, the relative truncation errors decrease from $\mathcal{O}(10^{-4})$ with four partitions, to $\mathcal{O}(10^{-5})$ when eight partitions are used.

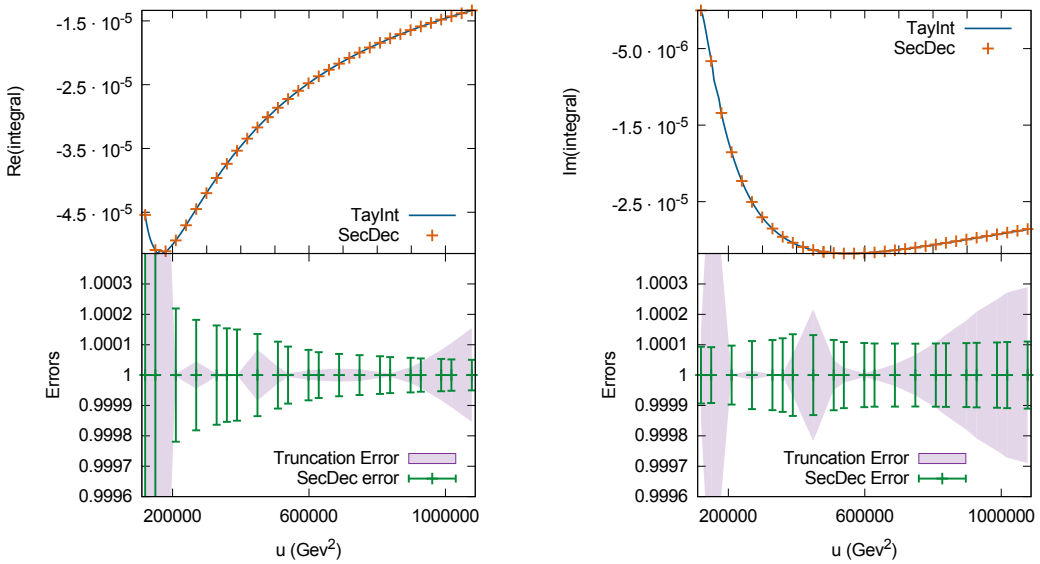


Figure 13.5: I10 over its threshold, calculated at $\mathcal{O}(\epsilon^0)$ with four orders and a high-uniform partition set for each subsector, choosing $m_2 = \frac{1}{\sqrt{2}} m_1$ and $m_1 = 173$ GeV. The lower plots show the relative uncertainties of the TayInt approximations and numerical SecDec results, respectively.

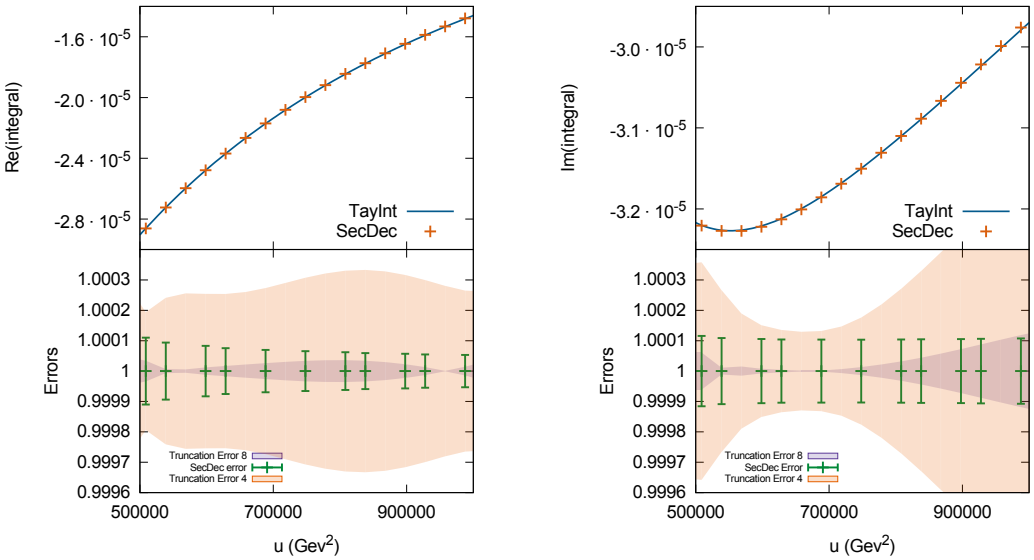


Figure 13.6: I10 calculated at $\mathcal{O}(\epsilon^0)$ with a fourth-order Taylor expansion. The scale u is over its $4m_1^2$ threshold, with the near threshold region excluded and $m_2 = \frac{1}{\sqrt{2}}m_1$, $m_1 = 173$ GeV. The lower plots show the relative TayInt, with four and eight partitions and SecDec uncertainties, respectively.

A more thorough quantitative analysis of the impact of the partitioning is given in Table 13.1, where the relative truncation error for the integral I10 is shown for different expansion orders and integral partitions. On the one hand, it shows that the accuracy increase by doubling the number of partitions is roughly equivalent to raising the order of the expansion by two. On the other hand, each doubling of the number of partitions leads to an order of magnitude gain in precision, demonstrating how the use of partitions achieves objective O2 of the final TayInt algorithm.

Table 13.1: The impact on the mean relative TayInt truncation error of changing the order of the Taylor expansion and the number of partitions in the TayInt algorithm applied to I10 at $\mathcal{O}(\epsilon^0)$. The kinematic region over which the mean is taken is given by $u \in [16m_1^2, 32m_1^2] = [467864, 957728]$ GeV^2, $m_2 = \frac{1}{\sqrt{2}}m_1$ and $m_1 = 173$ GeV.

Mean relative TayInt truncation error				
Number of Partitions	4		8	
Order	**Re(I10)**	**Im(I10)**	**Re(I10)**	**Im(I10)**
0	0.530165	0.623989	0.0812167	0.242449
2	0.0221554	0.0242271	0.000642405	0.00237282
4	0.00278254	0.00242541	0.000163342	0.000079292
6	0.000284179	0.000281809	0.0000239721	0.000038864

Close to thresholds, there can be occasional rapid changes at the end points of integration regions which lead to larger truncation errors. For example, increases of 10 % are observed for the $\mathcal{O}(\epsilon^0)$ coefficient of I10. Beyond $u = 30\,m_1^2$ in the kinematic region over the threshold, the points of rapid fluctuation move closer to the boundary of the integration region, however do not enter it. This also leads to a small reduction in the precision of the TayInt approximations, however this loss is at the 0.01% level for the $\mathcal{O}(\epsilon^0)$ coefficient of I10, demonstrating the realisation of objective O1 of the final TayInt algorithm. This is illustrated in Fig. 13.7 by plotting the absolute value of the first subsector of I10 near to the threshold, reasonably over the threshold and very far over the threshold in u.

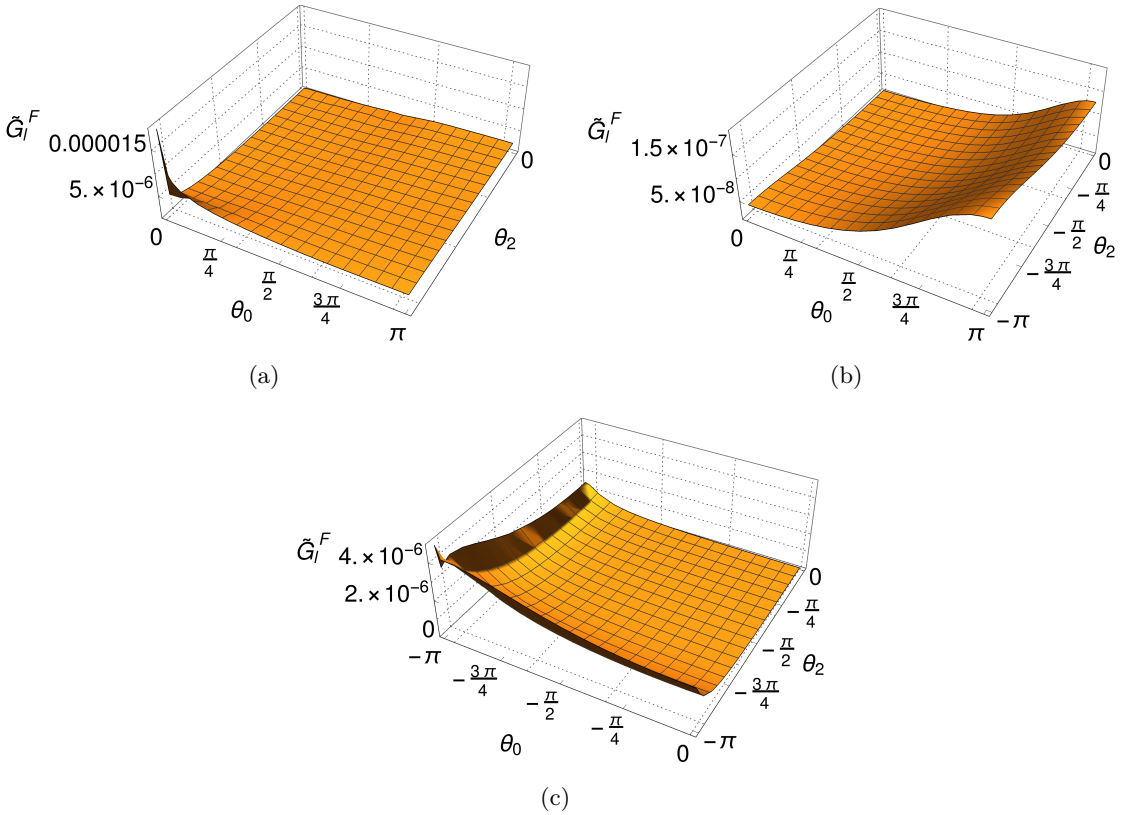


Figure 13.7: The absolute value of the first subsector of I10 at $\mathcal{O}(\epsilon^0)$ after step OT3 in the final TayInt algorithm plotted at; (a) a near threshold point $u = 179574$ GeV^2, (b) a point a reasonable distance over the threshold $u = 748225$ GeV^2, (c) a point very far over the threshold, $u = 1017586$ GeV^2. In all cases, $m_2 = \frac{1}{\sqrt{2}}m_1$ and $m_1 = 173$ GeV.

In Fig. 13.8 the fourth-order TayInt $\mathcal{O}(\epsilon^1)$ approximation with high-uniform partitions for the integral I10 is compared to the SecDec result in the over-threshold region, showing no drop in accuracy with respect to the lower order in ϵ, a realisation of objective O1 of the final TayInt algorithm. The SecDec results were computed using a relative accuracy of 10^{-3} with default numerical integration parameters and the integrator Vegas.

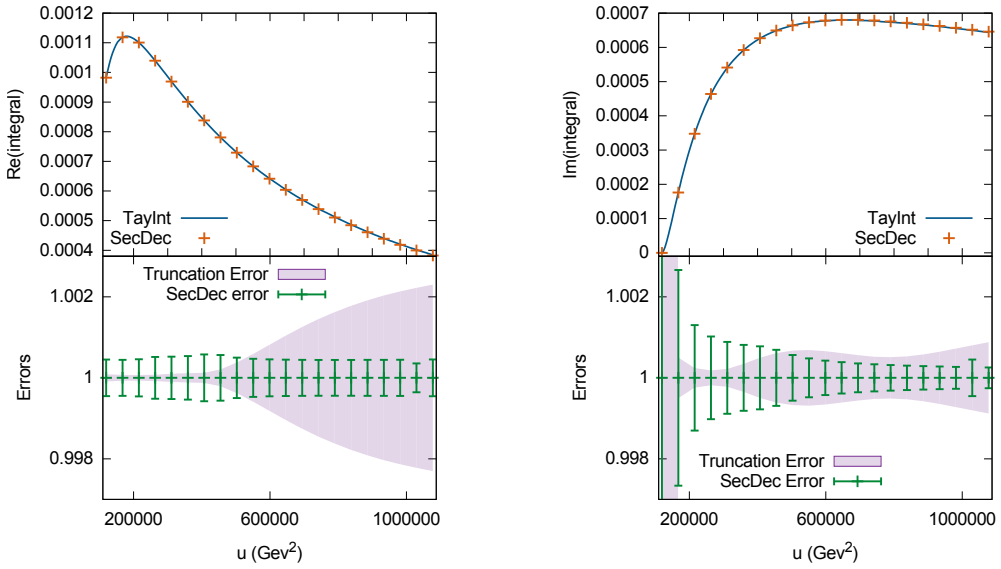


Figure 13.8: The I10 Integral calculated at $\mathcal{O}(\epsilon^1)$ with a fourth-order Taylor expansion and a high-uniform partition set for each subsector. The scale u is over its $4m_1^2$ threshold, with $m_2 = \frac{1}{\sqrt{2}}m_1$ and $m_1 = 173$ GeV. The lower plots show the relative uncertainties of the TayInt approximations and numerical SecDec results, respectively.

In Fig. 13.9, the $\mathcal{O}(\epsilon^2)$ TayInt approximation for the integral I10 is compared to the SecDec result in the over threshold region, again without diminished accuracy with respect to the lower orders in ϵ (realising objective O1). The approximations shown are based on a fourth-order Taylor expansion with a high-uniform partition set for each subsector. A relative accuracy of 10^{-4} was used for the production of the SecDec results.

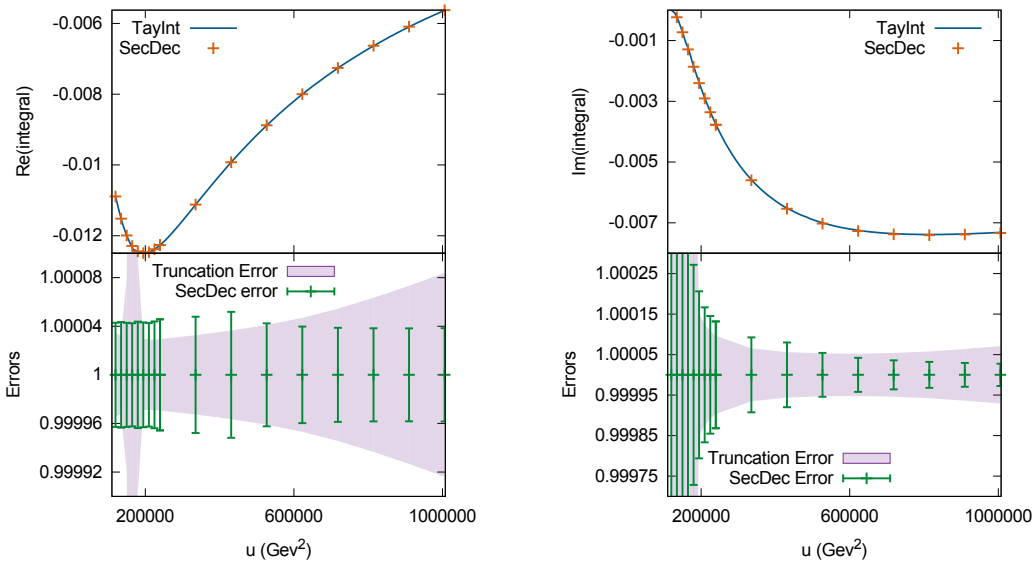


Figure 13.9: The I10 Integral calculated at $\mathcal{O}(\epsilon^2)$ with a fourth-order Taylor expansion and a high-uniform partition set for each subsector. The scale u is over its $4m_1^2$ threshold, with $m_2 = \frac{1}{\sqrt{2}}m_1$ and $m_1 = 173$ GeV. The lower plots show the relative TayInt and SecDec errors, respectively.

To make the behaviour of the uncertainty estimates in the near threshold region clearer, more points are plotted in the $u \in [4m_1^2, 6m_1^2]$ region. It is important to observe that the $\mathcal{O}(\epsilon^0)$ and $\mathcal{O}(\epsilon^2)$ TayInt approximations for I10 of Fig. 13.5 and 13.9, each display a region (well above the threshold) with an apparent deterioration of the TayInt truncation error. This is not a result of the Taylor expansion having a smaller radius of convergence, as is the case near or well above the threshold. Rather, this is a parametric effect that arises from two of the subsector integrands exhibiting oscillatory behaviour around zero in the vicinity of these kinematic points. These result in enhanced numerical cancellations among consecutive orders of the expansion. As a consequence, the uncertainty is grossly overestimated by the truncation error.

In Fig. 13.10 the TayInt approach is applied to the $\mathcal{O}(\epsilon^0)$ coefficient of the integral I39, with more propagators and scales, without a loss in accuracy compared to the simpler examples discussed above (realising objective O1). For the calculation a fourth-order Taylor expansion and a high-uniform partition set for each subsector were used. The TayInt approximations agree well with the SecDec results, and are stable over the whole kinematic region, given the truncation error only ever varies by 0.01% (realising objective O3). For the SecDec results a relative accuracy of 10^{-3} was chosen.

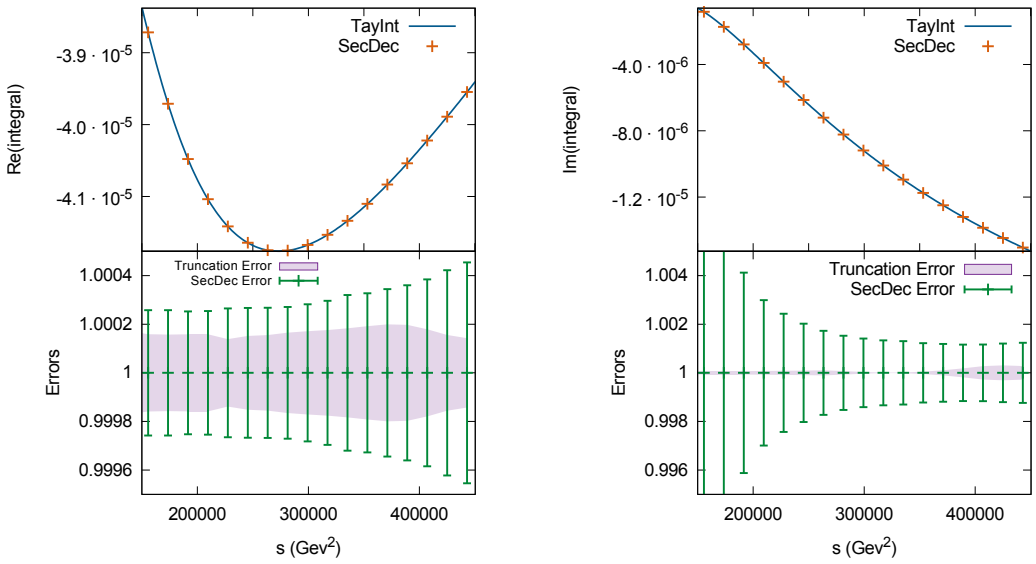


Figure 13.10: The I39 Integral calculated at $\mathcal{O}(\epsilon^0)$ with a fourth-order Taylor expansion and a high-uniform partition set for each subsector. The scale s is over the $4m_1^2$ threshold, with $u = -59858$ GeV^2, $m_2 = \frac{1}{\sqrt{2}}m_1$ and $m_1 = 173$ GeV. The lower plots show the relative TayInt and SecDec uncertainties, respectively.

In Fig. 13.11, the TayInt approach is applied to the $\mathcal{O}(\epsilon^1)$ coefficient of the integral I39. A fourth-order Taylor expansion and and a high-uniform partition set were used for each subsector. The TayInt approximations are consistent with the SecDec results across their full kinematic range (realising objective O3). To put this into context, the TayInt approximation, $0.000934066 + 0.000126179$i, at the ninth kinematic point at which SecDec is evaluated has an associated absolute uncertainty of $3.5495 \cdot 10^{-6} + 3.48459 \cdot 10^{-6}$i. The SecDec result at this point is $0.000933183 + 0.000127422$i. Thus there is agreement between TayInt and SecDec within the TayInt uncertainty. For the sake of comparison, the SecDec results for Fig. 13.11 were obtained asking for a relative accuracy of 10^{-3}.

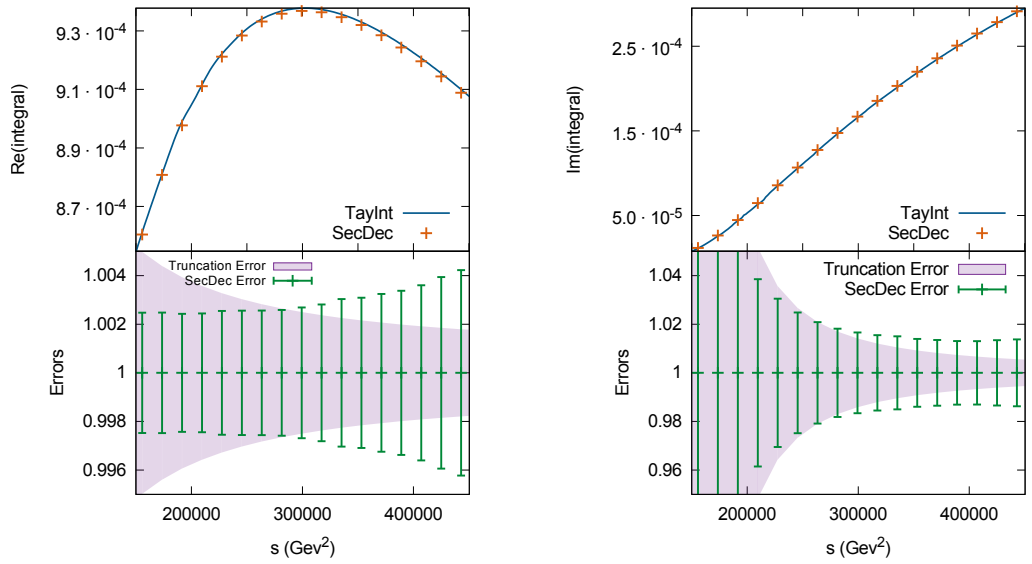


Figure 13.11: The I39 Integral calculated at $\mathcal{O}(\epsilon^1)$ with a fourth-order Taylor expansion and a high-uniform partition set for each subsector. The scale s is over the $4m_1^2$ threshold, with $u = -59858$ GeV^2, $m_2 = \frac{1}{\sqrt{2}}m_1$ and $m_1 = 173$ GeV. The lower plots show the relative TayInt and SecDec uncertainties, respectively.

In Fig. 13.12, the $\mathcal{O}(\epsilon^2)$ coefficient of the integral I39 is calculated with TayInt. A fourth-order Taylor expansion and a high-uniform partition set were used for each subsector. The algebraic TayInt approximation coincides with the SecDec results at the set of points considered above the $4m_1^2$ threshold. The $\mathcal{O}(\epsilon^2)$ SecDec results were obtained at a requested relative accuracy of 10^{-3}. It is already apparent from Fig. 13.3 that the accuracy of the approximations obtained for I39 below $s = 4m_1^2$ is independent of the order in ϵ and this is now shown to be true over the $4m_1^2$ threshold as well for I39, an integral for which there is, thus far, no analytic result. By its application to integrals with increasing numbers of scales and propagators, for different kinematic hierarchies and to order $\mathcal{O}(\epsilon^2)$, the predictive versatility of TayInt has been demonstrated, clearly showing how objectives O1-3 are satisfied.

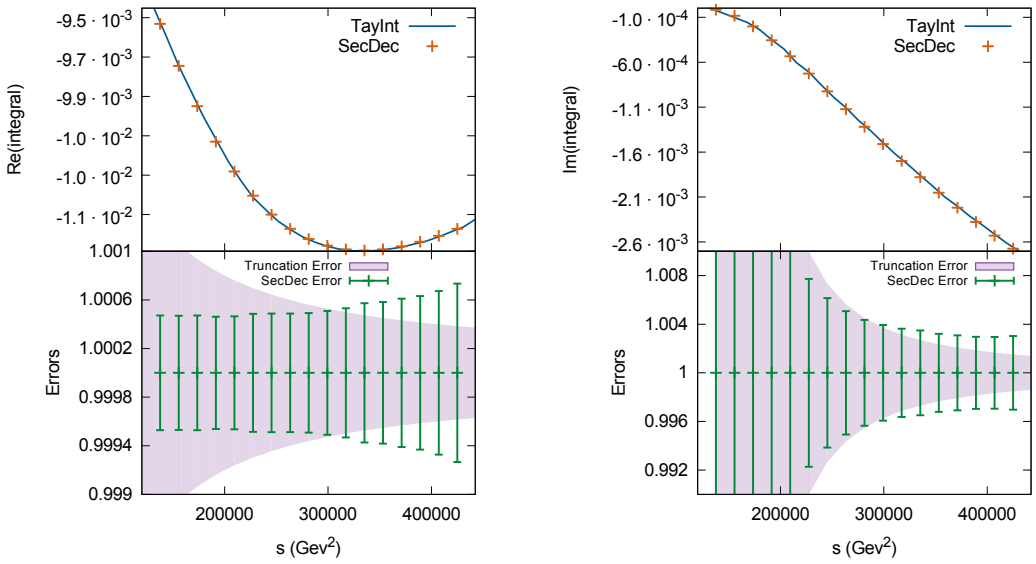


Figure 13.12: The I39 Integral calculated at $\mathcal{O}(\epsilon^2)$ with a fourth-order Taylor expansion and a high-uniform partition set for each subsector. The scale s is over the $4m_1^2$ threshold, with $u = -59858$ GeV^2, $m_2 = \frac{1}{\sqrt{2}}m_1$ and $m_1 = 173$ GeV. The lower plots show the relative TayInt and SecDec uncertainties, respectively.

In Fig. 13.13 the TayInt truncation errors obtained using eight and 16 partitions are plotted for the integral I39 at $\mathcal{O}(\epsilon^0)$ in yellow and purple bands, respectively. The y-axis is truncated so that the larger ratios near the threshold, due to the TayInt approximations and SecDec results being consistent with zero, cannot be seen. However, given the size of the SecDec errors in the imaginary part, an inset provides a closer look at the TayInt error bands. The TayInt approximation is based on a fourth-order Taylor expansion. The SecDec results have a relative accuracy of 10^{-3}. The TayInt uncertainties decrease by an order of magnitude when the number of partitions is doubled, replicating the effect seen for the I10 integral and generalising the realisation of objective O2. The average SecDec evaluation time at a kinematic point increases by a factor of 1.6 for each order of magnitude raise in relative precision, while, due to the fact that the TayInt algorithm produces an algebraic integral library, analytic in the kinematic scales, the evaluation using the TayInt algebraic approximation is always instantaneous.

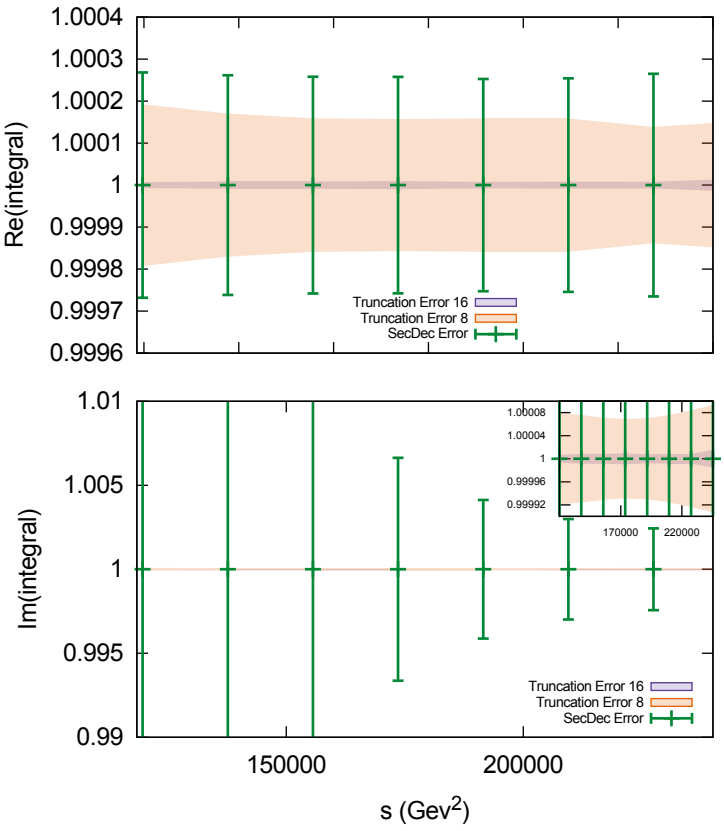


Figure 13.13: The relative TayInt uncertainty obtained with eight and 16 partitions respectively, for the integral I39 at $\mathcal{O}(\epsilon^0)$ above the threshold, with $u = -59858$ GeV^2, $m_2 = \frac{1}{\sqrt{2}} m_1$ and $m_1 = 173$ GeV.

In Fig. 13.14 the TayInt approximation for I21 at $\mathcal{O}(\epsilon^0)$ is plotted and compared to results from the program SecDec. The TayInt approximation is obtained with a fourth-order Taylor expansion and high-uniform, low-uniform and varied partition sets for each subsector, The SecDec results were computed using default numerical integration parameters and the integrator Vegas, asking for a relative accuracy of 10^{-4}. The plot shows the dependence on the scale u in the threshold region around $u = 4\,m_1^2 \sim 120000$ GeV^2 and above the threshold. By referring to Fig. 13.4(a), a smooth transition from the below-threshold expansion to the over-threshold expansion can be observed, demonstrating the achievement of objective O3 of the final TayInt algorithm. For the I10 integral, losses of precision were observed both close to the threshold and very far above it (illustrated in Fig. 13.7). A more pronounced rendering of the same effect is seen in Fig. 13.14. However, these deviations are within the TayInt uncertainty band. At

the third SecDec point the TayInt approximation with its uncertainties, corresponding lower and upper bound and the SecDec result are given respectively as:

- $-0.0000445 - 8.900 \cdot 10^{-6}\mathrm{i} \pm (2.386 \cdot 10^{-6} + 8.279 \cdot 10^{-7}\mathrm{i})$;
- $[-0.0000421 - 8.0726 \cdot 10^{-6}\mathrm{i}, -0.0000469 - 9.718 \cdot 10^{-6}\mathrm{i}]$;
- $-0.0000455 - 8.796 \cdot 10^{-6}\mathrm{i}$.

At the eleventh SecDec point the TayInt approximation with its uncertainties, corresponding lower and upper bound and the SecDec result are given respectively as:

- $-0.0000357 - 0.0000241\mathrm{i} \pm (2.751 \cdot 10^{-6} + 2.018 \cdot 10^{-6}\mathrm{i})$;
- $[-0.0000329 - 0.0000221\mathrm{i}, -0.0000384 - 0.0000261\mathrm{i}]$;
- $-0.0000333 - 0.0000245\mathrm{i}$.

Thus there is agreement between TayInt and SecDec within the TayInt uncertainty.

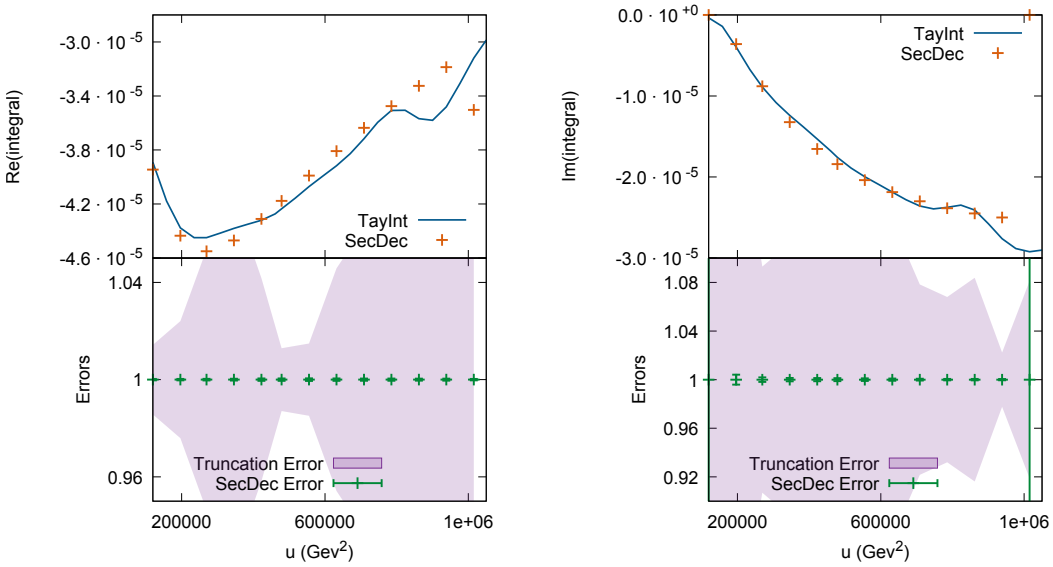


Figure 13.14: I21 over its threshold, calculated at $\mathcal{O}(\epsilon^0)$ with four orders and high-uniform, low-uniform and varied partition sets, choosing $m_2 = \frac{1}{\sqrt{2}} m_1$ and $m_1 = 173$ GeV. The lower plots show the relative uncertainties of the TayInt approximations and numerical SecDec results, respectively.

In Fig. 13.15 the fourth-order TayInt $\mathcal{O}(\epsilon^1)$ approximation with high-uniform partitions for the integral I21 is compared to the SecDec result in the over-threshold region,

showing no drop in accuracy with respect to the lower order in ϵ, a realisation of objective O1 of the final TayInt algorithm. The SecDec results were computed using a relative accuracy of 10^{-3} with default numerical integration parameters and the integrator Vegas. The same pattern of precision loss close to the threshold and very far above it that was observed for I21 at order ϵ^0 is seen in Fig. 13.15. However, these deviations are within the TayInt uncertainty band. At the sixth SecDec point the TayInt approximation with its uncertainties, corresponding lower and upper bound and the SecDec result respectively read:

- $0.000968 + 0.000337\mathrm{i} \pm (0.0000255 + 0.0000635\mathrm{i})$;
- $[0.000942 + 0.000274\mathrm{i}, 0.000993 + 0.000401\mathrm{i}]$;
- $0.000960 + 0.000359\mathrm{i}$.

And at the ninth SecDec point:

- $0.000868 + 0.000463\mathrm{i} \pm (0.0000405 + 0.0000446\mathrm{i})$;
- $[0.000828 + 0.000418\mathrm{i}, 0.000909 + 0.000508\mathrm{i}]$;
- $0.000859 + 0.000468\mathrm{i}$.

Thus there is agreement between TayInt and SecDec within the TayInt uncertainty.

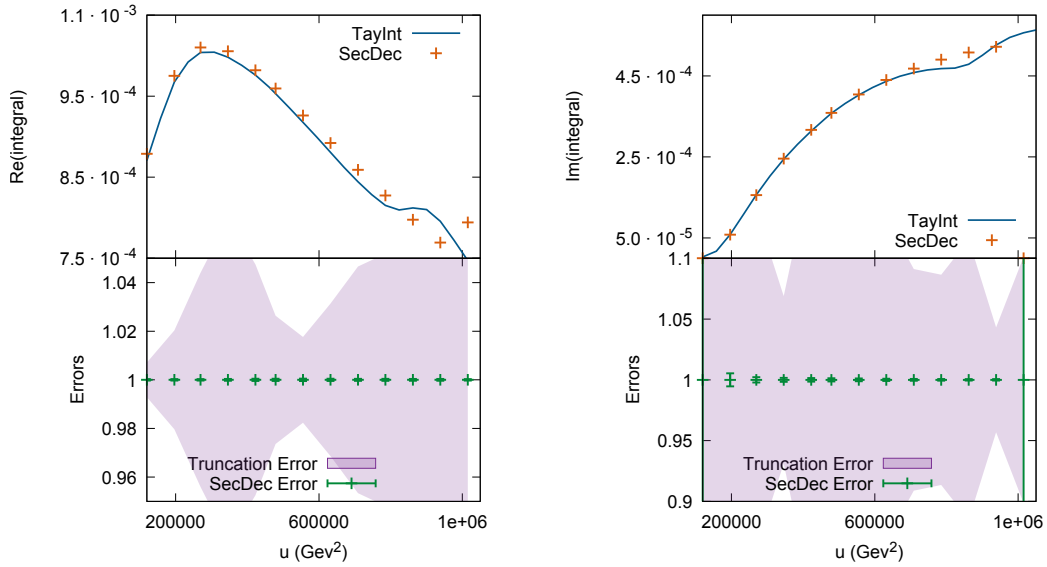


Figure 13.15: The I21 Integral calculated at $\mathcal{O}(\epsilon^1)$ with a fourth-order Taylor expansion and high-uniform, low-uniform and varied partition sets. The scale u is over its $4m_1^2$ threshold, with $m_2 = \frac{1}{\sqrt{2}}m_1$ and $m_1 = 173$ GeV. The lower plots show the relative uncertainties of the TayInt approximations and numerical SecDec results, respectively.

In Fig. 13.16, the $\mathcal{O}(\epsilon^2)$ TayInt approximation for the integral I21 is compared to the SecDec result in the over threshold region. Again, there is no reduction in accuracy with respect to the lower orders in ϵ (realising objective O1). The approximations shown are based on a fourth-order Taylor expansion with high-uniform, low-uniform and varied partition sets for each subsector. A relative accuracy of 10^{-4} was used for the production of the SecDec results. Once again, the pattern of precision loss close to and very far above the threshold is seen in Fig. 13.16. As there is no turning point far above the threshold, the precision loss there is significantly less as compared to that seen in Figs. 13.14-13.15 and resembles the loss exhibited by the I10 integral in Fig. 13.5. Once again, these deviations are well within the TayInt uncertainty band. At the fourth SecDec point the TayInt approximation with its uncertainties, corresponding lower and upper bound and the SecDec result respectively read:

- $-0.0115 - 0.00228\text{i} \pm (0.0000433 + 0.00009977\text{i})$;
- $[-0.0116 - 0.00238\text{i}, -0.0115 - 0.00218\text{i}]$;
- $-0.0114 - 0.00228\text{i}$.

At the eleventh SecDec point:

- $-0.00963 - 0.00533\text{i} \pm (0.0000440 + 0.0000678067\text{i})$;
- $[-0.00967 - 0.00540\text{i}, -0.00960 - 0.00526\text{i}]$;
- $-0.00960 - 0.00526\text{i}$.

Thus there is agreement between TayInt and SecDec within the TayInt uncertainty.

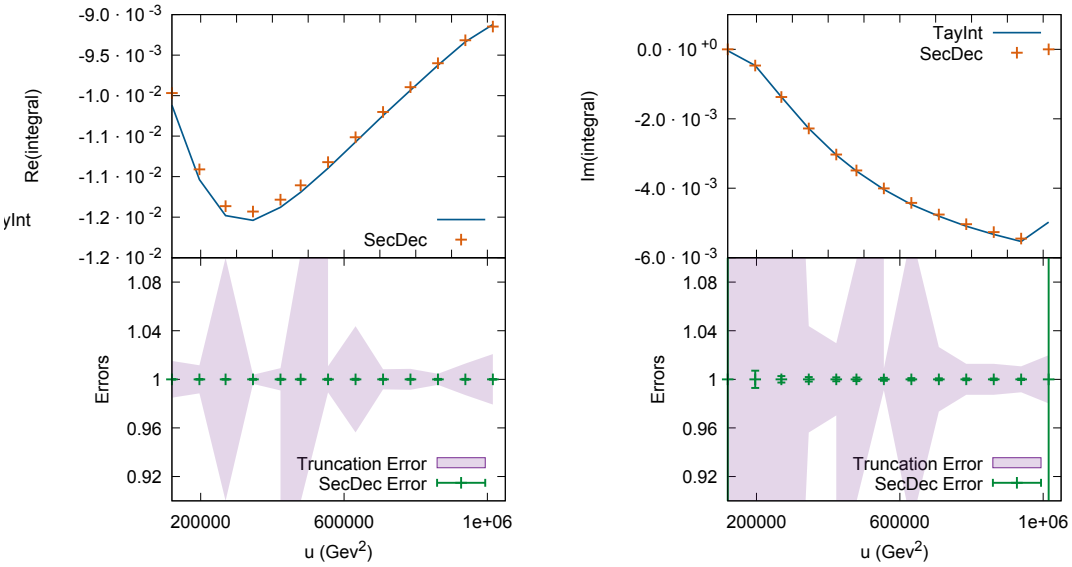


Figure 13.16: The I21 Integral calculated at $\mathcal{O}(\epsilon^2)$ with a fourth-order Taylor expansion and high-uniform, low-uniform and varied partition sets. The scale u is over its $4m_1^2$ threshold, with $m_2 = \frac{1}{\sqrt{2}}m_1$ and $m_1 = 173$ GeV. The lower plots show the relative TayInt and SecDec errors, respectively.

In the vicinity of the threshold and at very large values of the scale u, the I21 integral passes displays turning points. Turning points are difficult to descibe with a Taylor expansion as they correspond to several of the subsectors of I21 having a large derivative. This is the reason why the algebraic TayInt approximations for I21 evaluate less precisely numerically very far above the threshold as compared to I10 and I39. To demonstrate this effect more explicitly, the TayInt approximations are plotted for subsectors 3 and 5 of I21 at order ϵ^1 in Figs. 13.17-13.18. The former exhibits a turning point, whereas the latter does not. As can be seen, it is more difficult for the TayInt approximation to capture the behaviour of the turning point in Fig. 13.17. The SecDec

results are displayed alongside them, with the corresponding uncertainties featuring in the lower half of the plots.

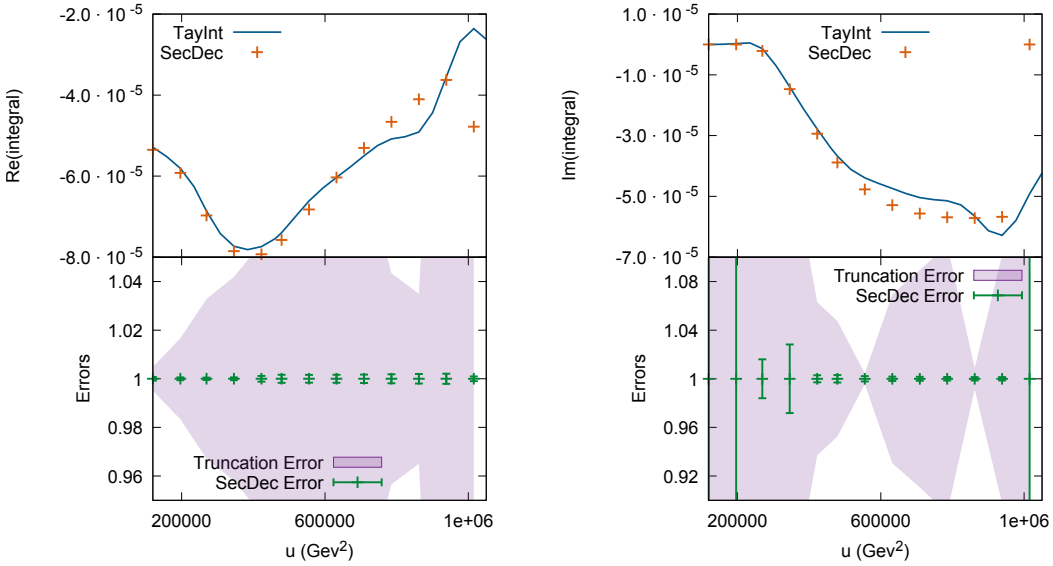


Figure 13.17: The third subsector of the I21 Integral calculated at $\mathcal{O}(\epsilon^1)$ with a fourth-order Taylor expansion and a varied partition set. The scale u is over its $4m_1^2$ threshold, with $m_2 = \frac{1}{\sqrt{2}}m_1$ and $m_1 = 173$ GeV. The lower plots show the relative uncertainties of the TayInt approximations and numerical SecDec results, respectively. This plot displays the turning point of the I21 integral at $u \sim 9m_1^2 = 269361$ Gev^2 and $u \sim 16m_1^2 = 478864$ GeV^2.

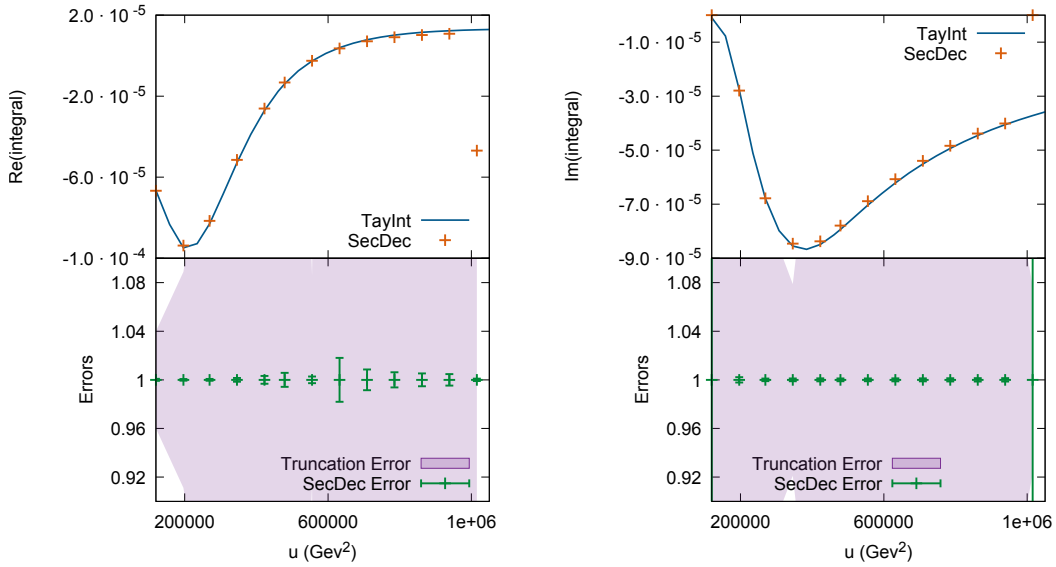


Figure 13.18: The fifth subsector of the I21 Integral calculated at $\mathcal{O}(\epsilon^1)$ with a fourth-order Taylor expansion and a high-uniform partition set. The scale u is over its $4m_1^2$ threshold, with $m_2 = \frac{1}{\sqrt{2}}m_1$ and $m_1 = 173$ GeV. The lower plots show the relative uncertainties of the TayInt approximations and numerical SecDec results, respectively. This plot displays the turning point of the I21 integral at $u \sim 9m_1^2 = 269361$ Gev^2 and $u \sim 16m_1^2 = 478864$ GeV^2.

To summarise the different causes of precision loss in the numerical evaluation of the algebraic TayInt approximations, the fifth subsector of I21 at order ϵ^2 is depicted in Fig. 13.19. The different forms of precision loss are:

1. Complete breakdown of the Taylor expansion at a threshold singularity. This takes the form of an explosion in the TayInt uncertainty. This is demonstrated by the uncertainty in the vicinity of $u \sim$ 119716 GeV^2 in the lower left panel in Fig. 13.19.

2. Slight degeneration of the Taylor expansion at very large values of the kinematic scales due to singularities approaching the integration region. This is characterised by a gradual rise in the TayInt uncertainty, shown in the lower left panel of Fig. 13.19 in the region $u \in [6 \cdot 10^5, 1 \cdot 10^6]$ GeV^2.

3. Loss of precision in the Taylor expansion in the presence of turning points and changes of sign. Due to the difficulty of describing zero or large derivatives with a Taylor expansion, this is identified by an abrupt spike in the TayInt uncertainty.

Examples are shown in the lower right panel of Fig. 13.19 in the vicinity of $u \sim 2 \cdot 10^5$ GeV^2 and $u \in [6 \cdot 10^5, 1 \cdot 10^6]$ GeV^2.

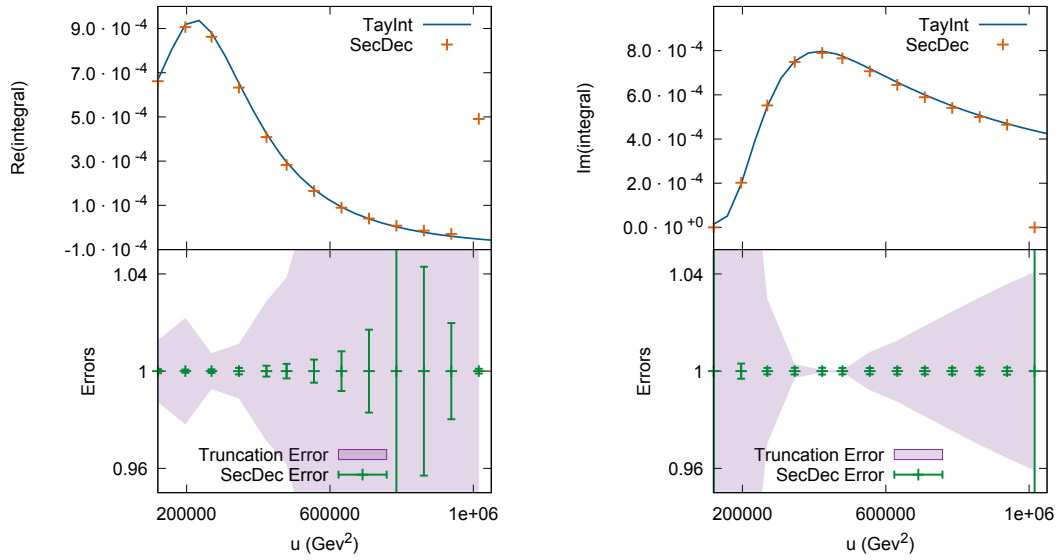


Figure 13.19: The fifth subsector of the I21 Integral calculated at $\mathcal{O}(\epsilon^2)$ with a fourth-order Taylor expansion and a high-uniform partition set. The scale u is over its $4m_1^2$ threshold, with $m_2 = \frac{1}{\sqrt{2}}m_1$ and $m_1 = 173$ GeV. The lower plots show the relative uncertainties of the TayInt approximations and numerical SecDec results, respectively. This plot displays the different kinds of precision losses that occur when calculating two-loop Feynman integrals with TayInt.

The mean difference between the TayInt approximations and SecDec results for I10, I21 and I39 up to order ϵ^2 over their threshold, Δ, is tabulated in Table 13.2. This reinforces how the objectives O1-3 are satisfied for a variety of different two-loop Feynman integrals by utilising various features of the TayInt algorithm. The TayInt approximations for I10 and I39 are based on a fourth-order, high-uniform partition, Taylor expansion. This is because the contour configuration alone guarantees the accuracy of the approximation and the partitioning can be raised arbitrarily to raise the precision. Hence only steps U1-OT7 are required in the final TayInt algorithm. But, for I21, the presence of points of sustained breakdown of the Taylor expansion makes an accurate approximation more difficult to generate. The partition set is also needed to produce an accurate and precise algebraic approximation for I21. Thus, the partitioning cannot be arbitrarily raised to increase the precision of the approximation. The algebraic approximations for I10 are 112.4 MB in size. For I21 they occupy 889.6 MB, while for I39 they

Table 13.2: The mean difference Δ (over epsilon orders ϵ^0-ϵ^2) between TayInt and SecDec, normalised to the SecDec result. The kinematic points are $u \in [4m_1^2, 36m_1^2] = [119716, 1077444]$ GeV^2 for I10 and I21 and $s \in [4m_1^2, 16m_1^2] = [119716, 478864]$ GeV^2 for I39.

Graph	Algorithm Steps Used	Re (Δ)	Im (Δ)
I10	U1-7	0.000658	0.000270
I21	U1-10	0.00220	0.00173
I39	U1-7	0.0000763	0.0000668

total 488.8 MB.

One of the most attractive features of the algebraic TayInt approximations is that the precision is independent of the ϵ order. Nevertheless the computation time, both for the configuration determination and the actual calculation, increases significantly when going to higher orders in ϵ.

13.3 Application to Divergent, Non-planar and Elliptic Two-loop Integrals

To illustrate the effect of implementing changes C1-3 to construct the final TayInt algorithm, in this section algebraic approximations obtained for divergent, non-planar and elliptic integrals, as depicted in Fig. 13.20, will be presented and their conformity to the objectives O1-3 remarked upon. To elaborate:

1. Algebraic approximations for the *elliptic* integral I59 are presented. This demonstrates that the mathematical complexity of the input integral does not affect the capacity of the final TayInt algorithm to produce an accurate algebraic approximation that evaluates to yield precise numerical approximations at arbitrary kinematic points.

> Upshot: TayInt is unaffected by the mathematical complexity of the input Feynman integral.

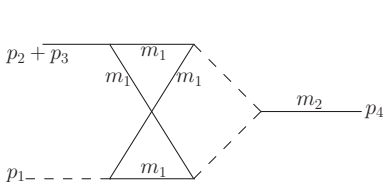
2. Algebraic approximations for the *non-planar* integral I246 are given. This reveals that the TayInt approach applies equally well to integrals with multiple thresholds and that the threshold-finding algorithm (step U2) is effective.

> Upshot: TayInt is unaffected by the number of thresholds of the input Feynman integral.

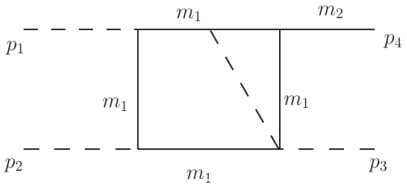
3. Algebraic approximations for the *divergent* integral I503 are given, crucially highlighting that TayInt can generate accurate and precise approximations *even when* a finite basis cannot be found.

Upshot: TayInt is unaffected by the whether or not a finite basis can be found for the input Feynman integral.

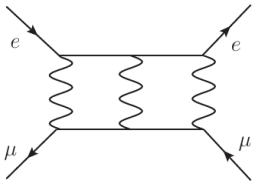
In what follows, these algebraic approximations are presented below and above any kinematic thresholds and for different orders in ϵ, respectively.



(a) I246



(b) I59



(c) I503

Figure 13.20: The graphs I246 (non-planar), (a), I59 (elliptic), (b) which contribute to Higgs-plus-jet at two-loop [32, 60, 127] and one of the two-loop divergent diagrams that contributes to μe scattering [132], I503, (c). Dashed lines indicate massless, solid internal lines massive and dots squared propagators. Solid external lines denote, where indicated, massive and else off-shell particles.

In Fig. 13.21 the TayInt approximations for the divergent coefficients of I503 in the Euclidean are plotted, providing the justification for removing the quasi-finite basis step from the final TayInt algorithm.

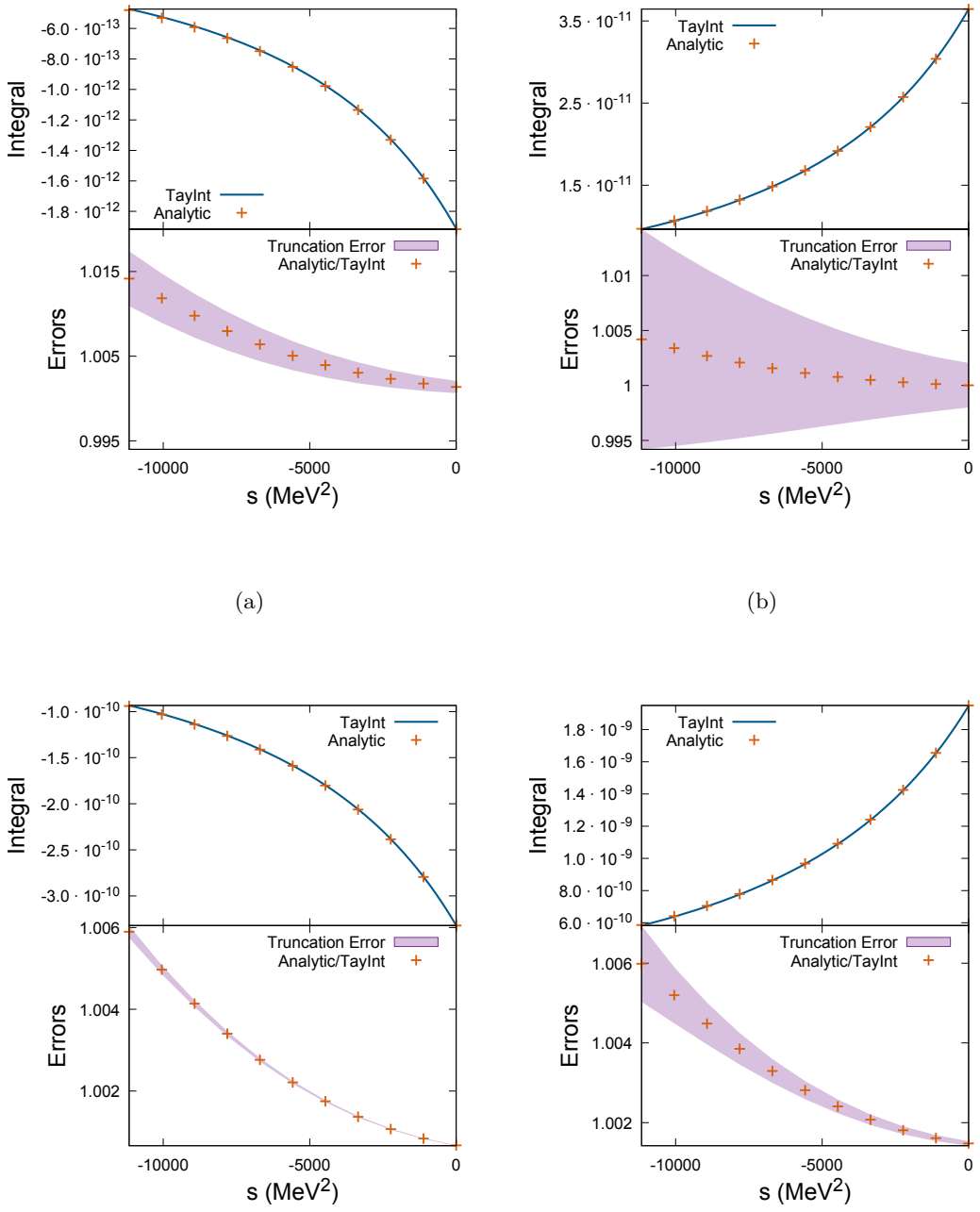


Figure 13.21: I503 in the Euclidean kinematic region, calculated with four orders and three partitions at; (a) $\mathcal{O}(\epsilon^{-4})$, (b) $\mathcal{O}(\epsilon^{-3})$, (c) $\mathcal{O}(\epsilon^{-2})$ and (d) $\mathcal{O}(\epsilon^{-1})$. The scales are $t = -0.5m_\mu^2, u = -0.5m_\mu^2$ and s running from 0 to $-m_\mu^2$, where $m_\mu^2 = 11166.6$ MeV^2, the muon mass squared, to ensure that $|s| + |t| + |u| <= 2m_\mu^2$. The lower plots show the relative uncertainties of the TayInt approximations.

In Figs. 13.22-13.23 the TayInt approximations for the triple-threshold $(0, 4m_1^2, 9m_1^2)$ non-planar integral I246 at order ϵ^0 are presented in the three over-threshold regions located by TayInt: $u \in [0, m_1^2]$; $u \in [m_1^2, 4m_1^2]$; $u \in [4m_1^2, 9m_1^2]$. As identical contour configurations and partition sets are for the first two such regions, these are shown on the same plot, Fig. 13.22. Due to the fact that the regions shown in Figs. 13.22-13.23 are bounded from above and below by threshold singularities, the Taylor expansion breaks down by construction in the vicinity of $u \sim 0$ GeV^2, $u \sim m_t^2 = 29929$ GeV^2, $u \sim 4m_t^2 = 119716$ GeV^2 and $u \sim 9m_t^2 = 269361$ GeV^2. However, the resulting deviations between TayInt and SecDec are accompanied by a corresponding increase in size of the TayInt uncertainty band.

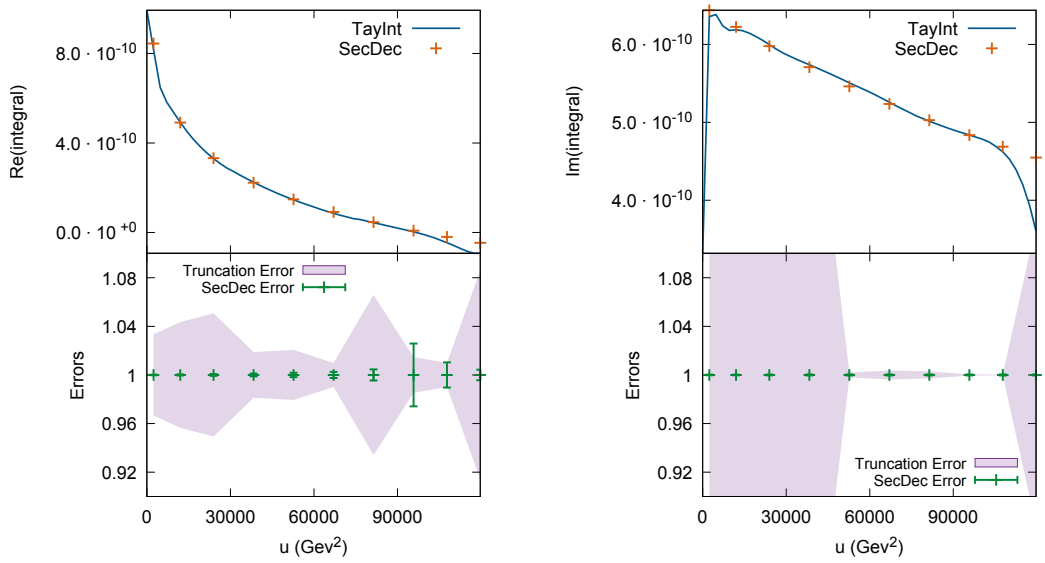


Figure 13.22: I246 in its first and second TayInt over-threshold regions, $u \in [0, 4m_1^2]$, calculated at $\mathcal{O}(\epsilon^0)$. A fourth-order Taylor expansion was used. High-uniform, low-uniform and varied partition sets were employed for each subsector. The kinematic scales are $m_2 = \frac{1}{\sqrt{2}} m_1$ and $m_1 = 173$ GeV. The lower plots show the relative uncertainties of the TayInt approximations and numerical SecDec results, respectively.

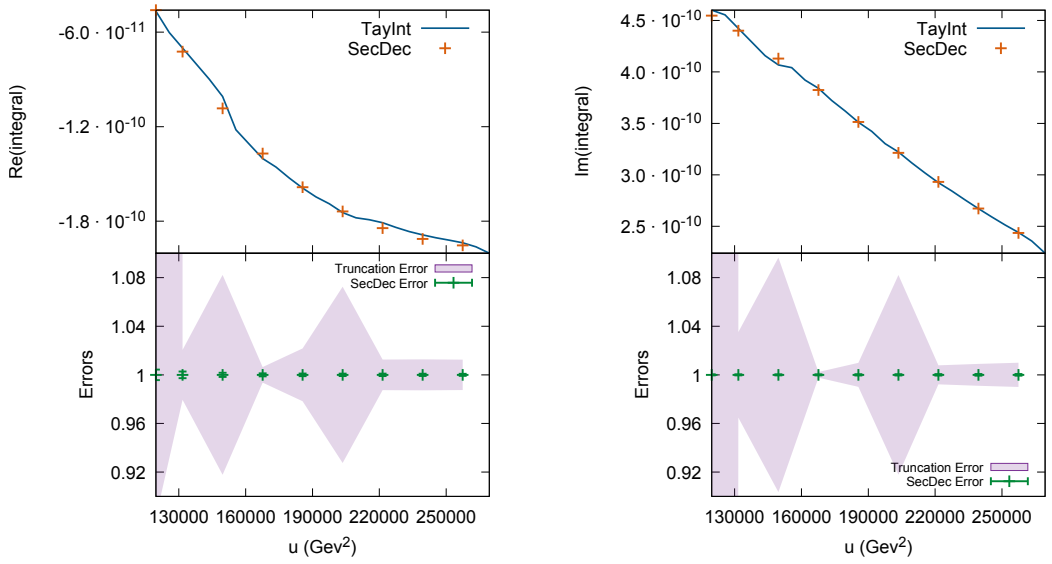


Figure 13.23: I246 in its third TayInt over-threshold region, $u \in [4m_1^2, 9m_1^2]$, calculated at $\mathcal{O}(\epsilon^0)$. A fourth-order Taylor expansion was used. High-uniform, low-uniform and varied partition sets were employed for each subsector. The kinematic scales are $m_2 = \frac{1}{\sqrt{2}} m_1$ and $m_1 = 173$ GeV. The lower plots show the relative uncertainties of the TayInt approximations and numerical SecDec results, respectively.

In Figs. 13.24-13.26 and 13.27-13.29 the TayInt approximations for the triple-threshold $(0,\ 4m_1^2,\ 9m_1^2)$ non-planar integral I246 at order ϵ^1 and ϵ^2 are presented. In this case different contour configurations and partition sets are chosen for all three over-threshold regions located by TayInt: $u \in [0, m_1^2]$; $u \in [m_1^2, 4m_1^2]$; $u \in [4m_1^2, 9m_1^2]$. They are shown using separate plots. These coefficients are maximally difficult to approximate algebraically with a Taylor expansion due to the presence of multiple threshold singularities, particularly visible for the ϵ^2 coefficient (Figs. 13.27-13.29). In particular, Figs. 13.25 and 13.28 demonstrate that in the vicinity of $u = m_1^2 = 29929$ GeV^2 the imaginary part of I246 is especially difficult to describe with a Taylor expansion. This justifies why the TayInt threshold-finding algorithm selects that point as a threshold, despite the fact that it is a pseudo-threshold. Moreover, the difference between the threshold points in Figs. 13.24-13.26 and Figs. 13.27-13.29 is small compared to the range over which I10 and I21 were plotted in Figs. 13.5-13.9 and Figs. 13.16-13.16. Thus, each plot of I246 is comparable to plotting the near-threshold region of I10 and I21 in increased detail. In fact it is worse, because of the presence of a threshold singularity at both ends of the kinematic ranges over which I246 is plotted. Because of this, the Taylor expansion

breaks down in a greater proportion of the kinematic range than in previous plots. This is characterised by the explosion of the TayInt uncertainty seen in the lower panels. The approximations shown in Figs. 13.22-13.29 demonstrate the ability of TayInt to provide accurate, precise and kinematically generalisable algebraic approximations even for Feynman integrals with multiple thresholds. These plots also depict the usefulness of the threshold-finding algorithm. At present no other kinematically algebraic results valid throughout phase space are available for the I246 integral in the literature. The precision of the approximations for I246 will now be discussed. In Fig. 13.24, at the second SecDec point the TayInt approximation with its uncertainties, corresponding lower and upper bound and the SecDec result respectively read:

- $-1.363 \cdot 10^{-8} - 1.243 \cdot 10^{-8}\mathrm{i} \pm (6.329 \cdot 10^{-10} + 4.328 \cdot 10^{-9}\mathrm{i})$;
- $[-1.426 \cdot 10^{-8} - 1.675 \cdot 10^{-8}\mathrm{i}, -1.300 \cdot 10^{-8} - 8.100 \cdot 10^{-8}\mathrm{i}]$;
- $-1.317 \cdot 10^{-8} - 1.090 \cdot 10^{-8}\mathrm{i}$.

At the ninth SecDec point:

- $-5.322 \cdot 10^{-9} - 1.152 \cdot 10^{-8}\mathrm{i} \pm (1.254 \cdot 10^{-9} + 4.973 \cdot 10^{-9}\mathrm{i})$;
- $[-6.577 \cdot 10^{-9} - 1.650 \cdot 10^{-8}\mathrm{i}, -4.068 \cdot 10^{-9} - 6.552 \cdot 10^{-9}\mathrm{i}]$;
- $-6.474 \cdot 10^{-9} - 1.100 \cdot 10^{-8}\mathrm{i}$.

Thus there is agreement between TayInt and SecDec within the TayInt uncertainty.

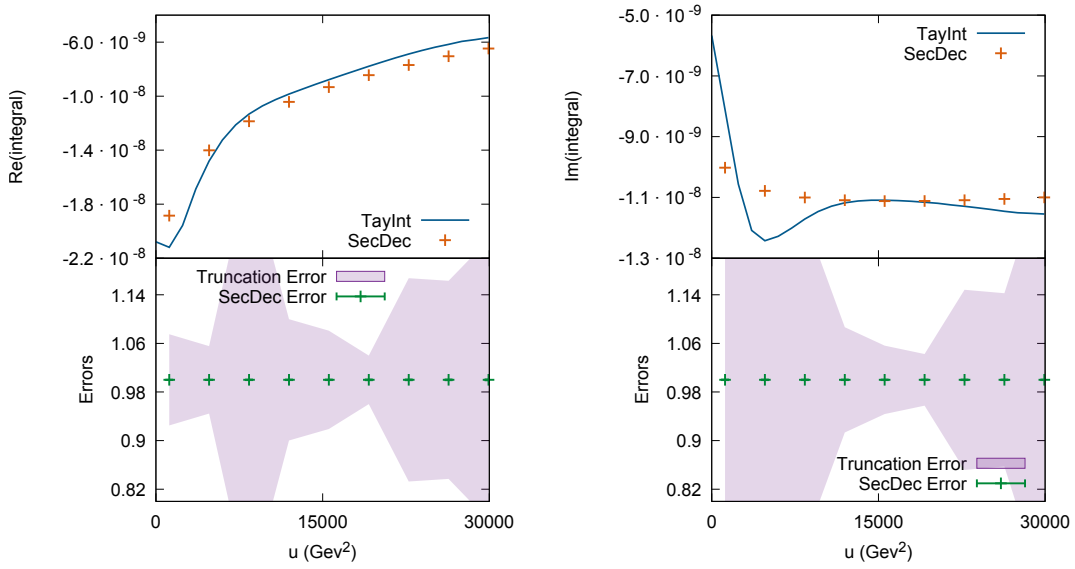


Figure 13.24: I246 in its first TayInt over-threshold region, $u \in [0, m_1^2]$, calculated at $\mathcal{O}(\epsilon^1)$. A fourth-order Taylor expansion was used. High-uniform, low-uniform and varied partition sets were employed for each subsector. The kinematic scales are $m_2 = \frac{1}{\sqrt{2}} m_1$ and $m_1 = 173$ GeV. The lower plots show the relative uncertainties of the TayInt approximations and numerical SecDec results, respectively.

Fig. 13.25 once again illustrates the difficulty in resolving a small region of phase space trapped between threshold singularities with a Taylor expansion. To be explicit, at the fourth SecDec point the TayInt approximation with its uncertainties, corresponding lower and upper bound and the SecDec result are given as follows:

- $-5.150 \cdot 10^{-9} - 1.005 \cdot 10^{-8}\mathrm{i} \pm (1.122 \cdot 10^{-9} + 8.028 \cdot 10^{-10}\mathrm{i})$;
- $[-6.273 \cdot 10^{-9} - 1.085 \cdot 10^{-8}\mathrm{i}, -4.028 \cdot 10^{-9} - 9.243 \cdot 10^{-9}\mathrm{i}]$;
- $-4.679 \cdot 10^{-9} - 1.074 \cdot 10^{-8}\mathrm{i}$.

At the penultimate SecDec point:

- $-7.178 \cdot 10^{-9} - 8.487 \cdot 10^{-9}\mathrm{i} \pm (5.122 \cdot 10^{-10} + 1.593 \cdot 10^{-9}\mathrm{i})$.;
- $[-7.682 \cdot 10^{-9} - 1.008 \cdot 10^{-8}\mathrm{i}, -6.658 \cdot 10^{-9} - 6.894 \cdot 10^{-9}\mathrm{i}]$;
- $-6.660 \cdot 10^{-9} - 9.282 \cdot 10^{-9}\mathrm{i}$.

Hence there is agreement between TayInt and SecDec within the TayInt uncertainty.

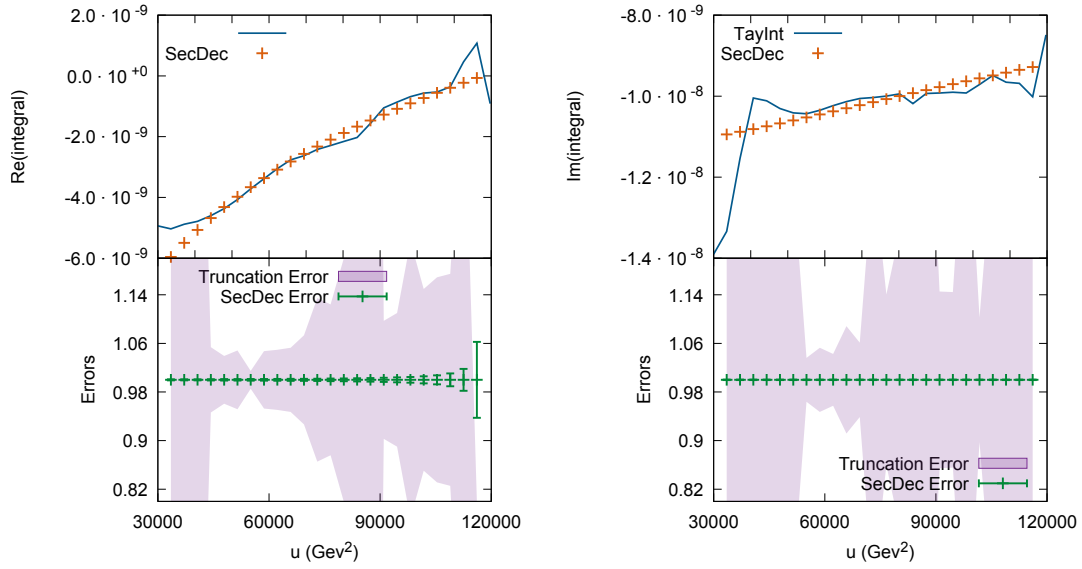


Figure 13.25: I246 in its second TayInt over-threshold region, $u \in [m_1^2, 4m_1^2]$, calculated at $\mathcal{O}(\epsilon^1)$. A fourth-order Taylor expansion was used. High-uniform, low-uniform and varied partition sets were employed for each subsector. The kinematic scales are $m_2 = \frac{1}{\sqrt{2}} m_1$ and $m_1 = 173$ GeV. The lower plots show the relative uncertainties of the TayInt approximations and numerical SecDec results, respectively.

In Fig. 13.26, at the second SecDec point the TayInt approximation with its uncertainties, corresponding lower and upper bound and the SecDec result read:

- $2.849 \cdot 10^{-10} - 9.950 \cdot 10^{-9}\mathrm{i} \pm (9.423 \cdot 10^{-11} + 1.481 \cdot 10^{-9}\mathrm{i})$;
- $[1.906 \cdot 10^{-10} - 1.143 \cdot 10^{-8}\mathrm{i}, 3.791 \cdot 10^{-10} - 8.469 \cdot 10^{-9}\mathrm{i}]$;
- $3.550 \cdot 10^{-10} - 9.100 \cdot 10^{-9}\mathrm{i}$.

At the last SecDec point:

- $3.828 \cdot 10^{-9} - 4.635 \cdot 10^{-9}\mathrm{i} \pm (1.095 \cdot 10^{-8} + 3.140 \cdot 10^{-7}\mathrm{i})$;
- $[-7.118 \cdot 10^{-9} - 3.186 \cdot 10^{-7}\mathrm{i}, 1.477 \cdot 10^{-8} + 3.093 \cdot 10^{-7}\mathrm{i}]$;
- $3.54441 \cdot 10^{-9} - 5.19699 \cdot 10^{-9}\mathrm{i}$.

So, there is agreement between TayInt and SecDec within the TayInt uncertainty.

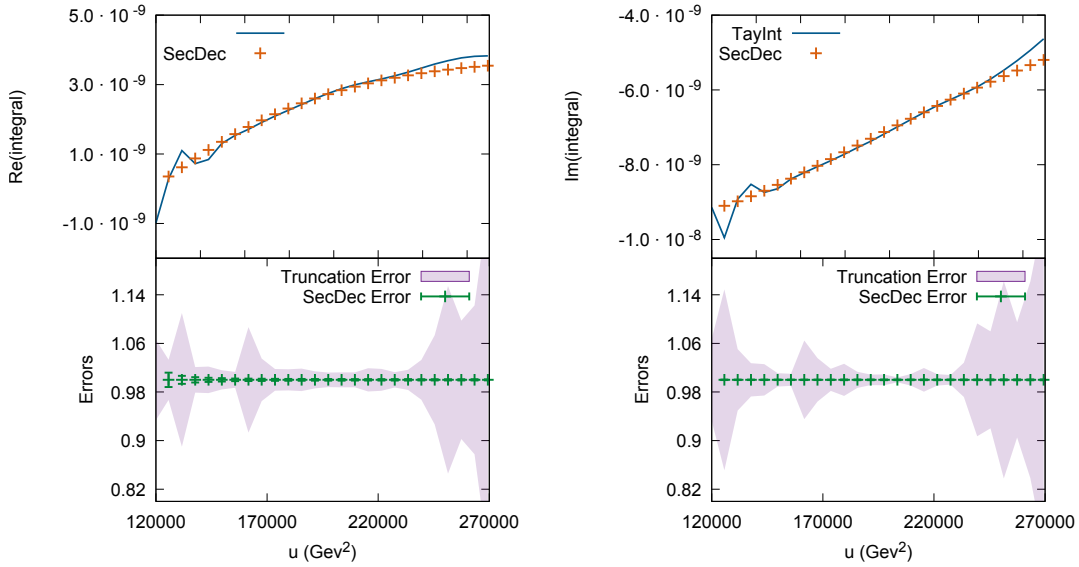


Figure 13.26: I246 in its third TayInt over-threshold region, $u \in [4m_1^2, 9m_1^2]$, calculated at $\mathcal{O}(\epsilon^1)$. A fourth-order Taylor expansion was used. High-uniform, low-uniform and varied partition sets were employed for each subsector. The kinematic scales are $m_2 = \frac{1}{\sqrt{2}}m_1$ and $m_1 = 173$ GeV. The lower plots show the relative uncertainties of the TayInt approximations and numerical SecDec results, respectively.

The ϵ^2 coefficient of I246 displays similar behaviour to its ϵ^1 counterpart in the region $u \in [0, 29929]$ GeV^2. There are once again losses of precision in the vicinity of the threshold singularities which bound the depicted kinematic region, the proximity to which over the relatively narrow kinematic range results in an explosion of the TayInt uncertainty. This is shown in Fig. 13.27, wherein, at the second SecDec point the TayInt approximation with its uncertainties, corresponding lower and upper bound and the SecDec result read:

- $1.541 \cdot 10^{-7} + 9.658 \cdot 10^{-8}\mathrm{i} \pm (2.860 \cdot 10^{-8} + 6.399 \cdot 10^{-9}\mathrm{i})$;
- $[2.221 \cdot 10^{-7} + 9.018 \cdot 10^{-8}\mathrm{i}, 2.793 \cdot 10^{-7} + 1.030 \cdot 10^{-7}\mathrm{i}]$;
- $1.401 \cdot 10^{-7} + 9.100 \cdot 10^{-8}\mathrm{i}$.

At the last SecDec point:

- $6.556 \cdot 10^{-8} + 9.759 \cdot 10^{-8}\mathrm{i} \pm (8.697 \cdot 10^{-9} + 4.021 \cdot 10^{-9}\mathrm{i})$;
- $[5.686 \cdot 10^{-8} + 9.357 \cdot 10^{-8}\mathrm{i}, 7.425 \cdot 10^{-8} + 1.016 \cdot 10^{-7}\mathrm{i}]$;

- $7.275 \cdot 10^{-8} + 1.031 \cdot 10^{-7}\mathrm{i}$.

Therefore, thee TayInt and SecDec values agree within the TayInt uncertainty.

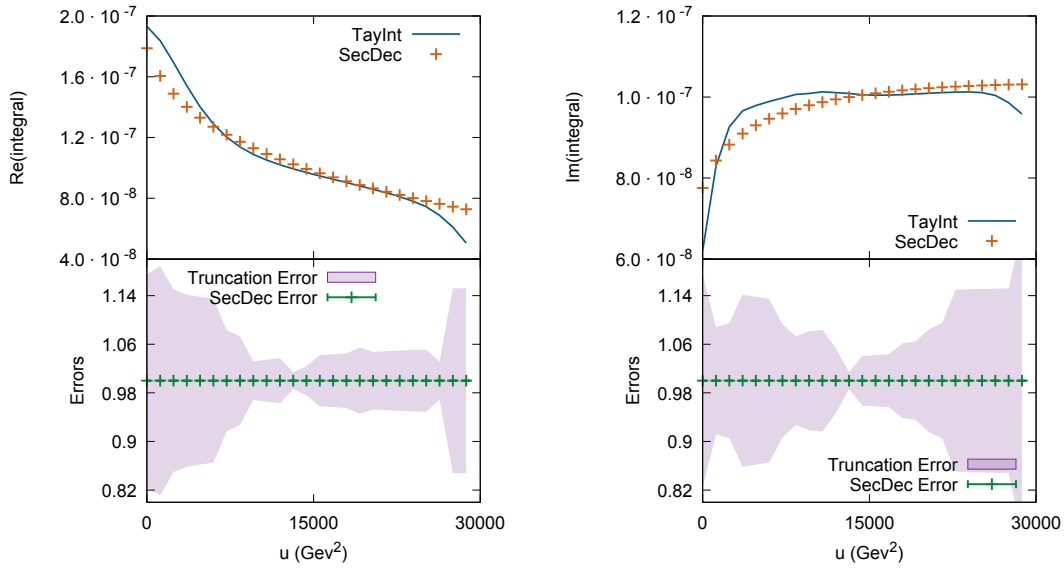


Figure 13.27: I246 in its first TayInt over-threshold region, $u \in [0, m_1^2]$, calculated at $\mathcal{O}(\epsilon^2)$. A fourth-order Taylor expansion was used. High-uniform, low-uniform and varied partition sets were employed for each subsector. The kinematic scales are $m_2 = \frac{1}{\sqrt{2}} m_1$ and $m_1 = 173$ GeV. The lower plots show the relative uncertainties of the TayInt approximations and numerical SecDec results, respectively.

The ϵ^2 coefficient of I246 also mirrors the behaviour shown by its ϵ^1 counterpart in the region $u \in [29929, 119716]$ Gev^2, as shown in Fig. 13.27. At the first SecDec point in Fig. 13.28, the TayInt approximation with its uncertainties, corresponding lower and upper bound and the SecDec result are:

- $6.582 \cdot 10^{-8} + 1.087 \cdot 10^{-7}\mathrm{i} \pm (7.093 \cdot 10^{-9} + 5.857 \cdot 10^{-9}\mathrm{i})$;
- $[5.872 \cdot 10^{-8} + 1.029 \cdot 10^{-7}\mathrm{i}, 7.291 \cdot 10^{-8} + 1.146 \cdot 10^{-7}\mathrm{i}]$;
- $7.275 \cdot 10^{-8} + 1.031 \cdot 10^{-7}\mathrm{i}$.

At the penultimate SecDec point the TayInt approximation is already breaking down because of proximity to the threshold, as can be seen by the explosion of the TayInt uncertainty.

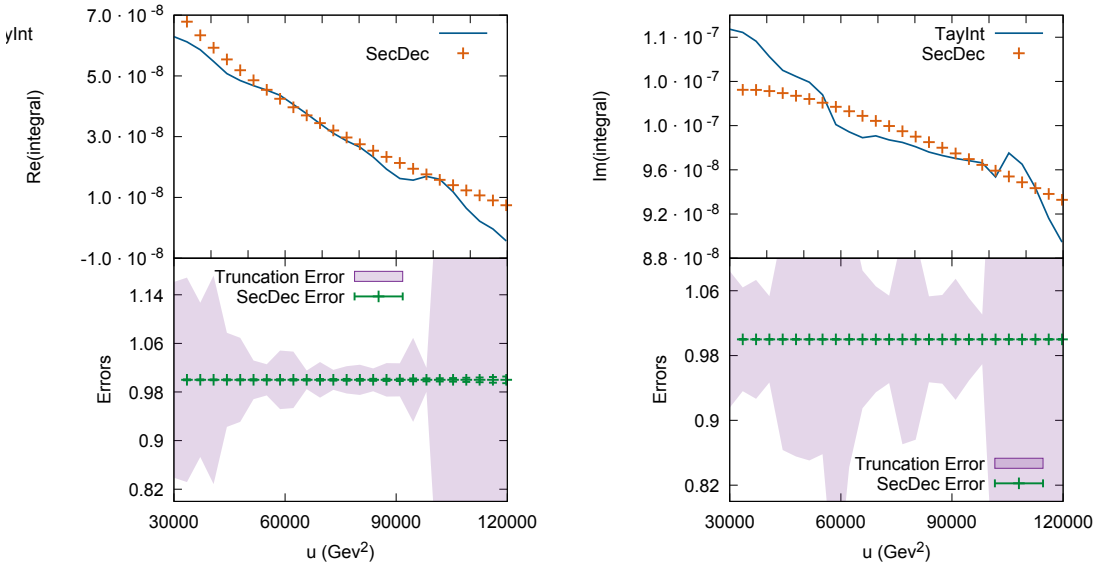


Figure 13.28: I246 in its second TayInt over-threshold region, $u \in [m_1^2, 4m_1^2]$, calculated at $\mathcal{O}(\epsilon^2)$. A fourth-order Taylor expansion was used. High-uniform, low-uniform and varied partition sets were employed for each subsector. The kinematic scales are $m_2 = \frac{1}{\sqrt{2}} m_1$ and $m_1 = 173$ GeV. The lower plots show the relative uncertainties of the TayInt approximations and numerical SecDec results, respectively.

Finally, at the 20th SecDec point in Fig. 13.29, the TayInt approximation with its uncertainties, corresponding lower and upper bound and the SecDec result are:

- $-2.851 \cdot 10^{-8} + 6.162 \cdot 10^{-8}\mathrm{i} \pm (1.917 \cdot 10^{-9} + 4.779 \cdot 10^{-9}\mathrm{i})$;
- $[-3.042 \cdot 10^{-8} + 5.684 \cdot 10^{-8}\mathrm{i}, -2.659 \cdot 10^{-8} + 6.640 \cdot 10^{-8}\mathrm{i}]$;
- $2.693 \cdot 10^{-8} + 6.669 \cdot 10^{-8}\mathrm{i}$.

This shows that the TayInt and SecDec values coincide within the TayInt uncertainty. To sum up, in the vicinity of the threshold singularities at the beginning and end of the kinematic range depicted in Fig. 13.29 the Taylor expansion breaks down. The losses of precision observed in Fig. 13.24-Fig. 13.29 that are *not* due to breakdown close to threshold are caused by the truncation of the Taylor expansion at fourth order. The TayInt program offers the user the option to raise the order of the Taylor expansion beyond this default setting if necessary.

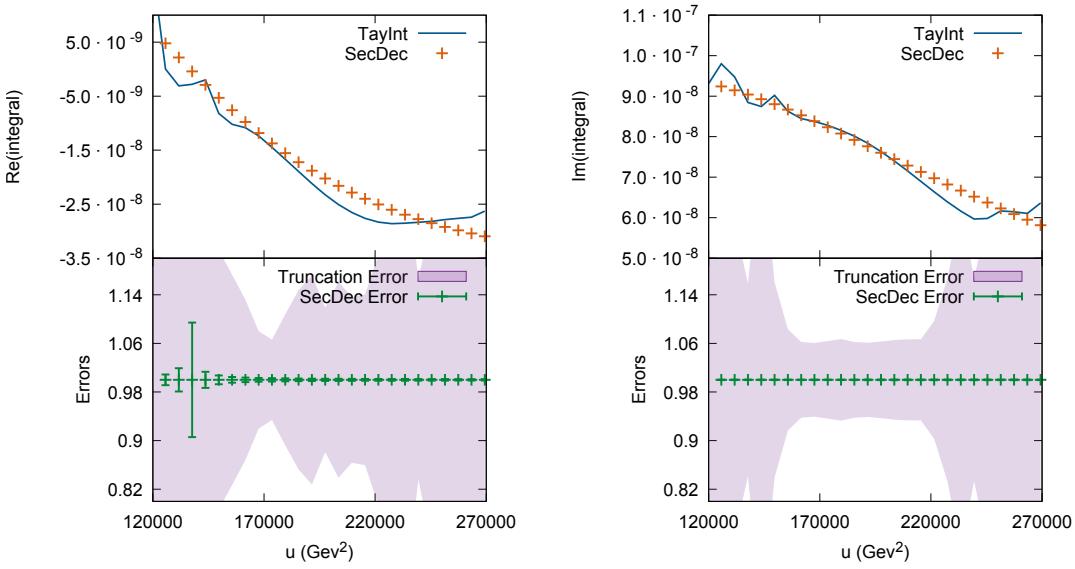
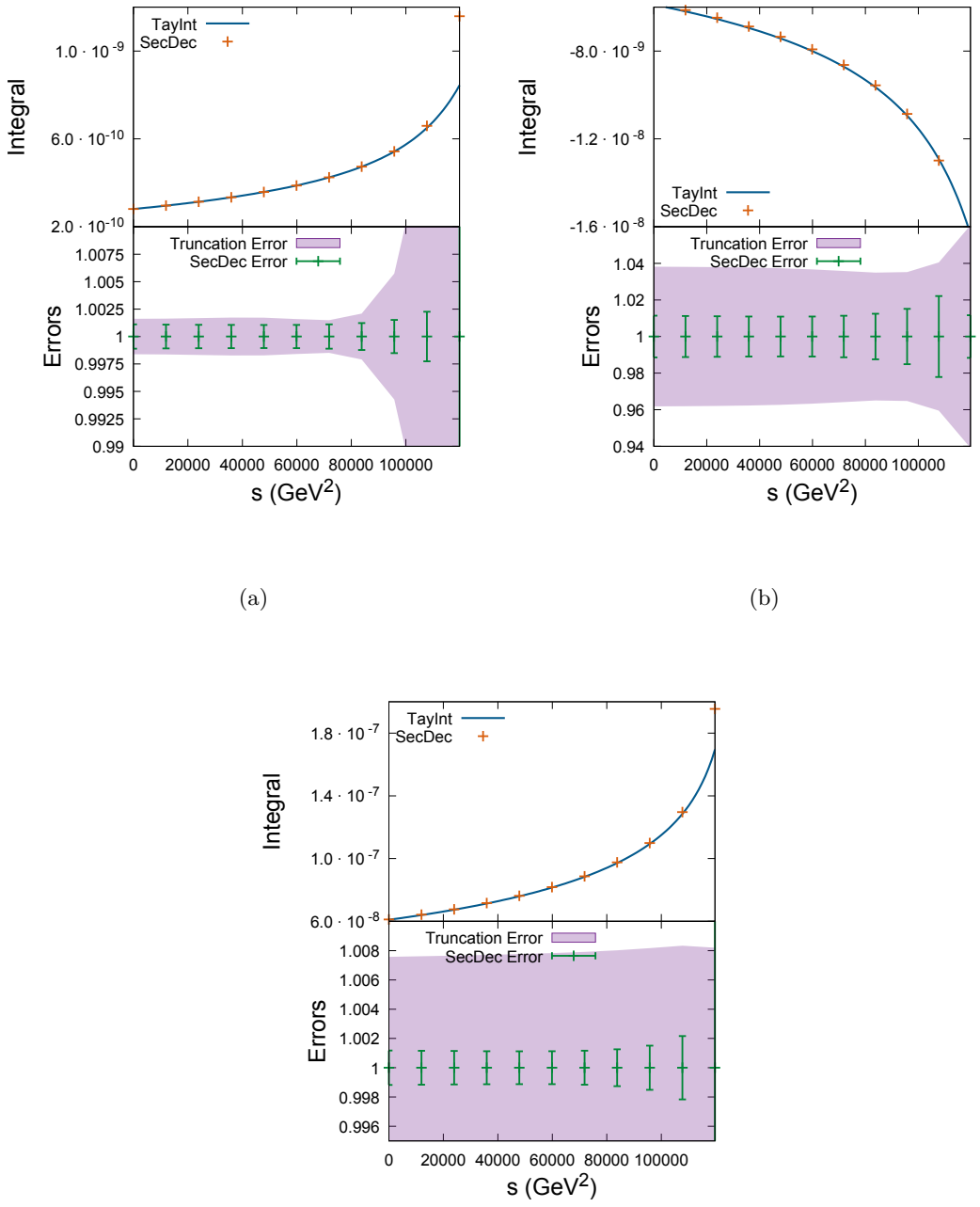


Figure 13.29: I246 in its third TayInt over-threshold region, $u \in [4m_1^2, 9m_1^2]$, calculated at $\mathcal{O}(\epsilon^2)$. A fourth-order Taylor expansion was used. High-uniform, low-uniform and varied partition sets were employed for each subsector. The kinematic scales are $m_2 = \frac{1}{\sqrt{2}}m_1$ and $m_1 = 173$ GeV. The lower plots show the relative uncertainties of the TayInt approximations and numerical SecDec results, respectively.

In Figs. 13.30-13.33 the TayInt approximations for the elliptic I59 integral are presented, representing the first arbitrarily valid, kinematically algebraic calculation of this Feynman integral. These approximations constitute the greatest success of TayInt. This is because they show that the program can produce accurate, precise and kinematically generalisable algebraic approximations even for an elliptic four-point two-loop integral, which represents the maximal level of contemporary difficulty for a Feynman integral. It also shows the worth of the final TayInt algorithm. Even for Feynman integrals with the greatest scarcity of suitable contours and maximal amount of fluctuation with respect to the integration parameters and kinematic scales, it can achieve the objectives O1-3 as intended. It is evident that the TayInt approximations for the integrals I236 and I59 (especially beyond leading order in ϵ), evaluate less smoothly numerically and exhibit greater deviation from SecDec than the simpler integrals I10 and I39, illustrating the stern mathematical complexity of these integrals. This is due to the multiple thresholds contained by I246 and the multiple turning points of I59. Such is the difficulty in producing algebraic approximations that achieve objectives O1-3 of TayInt for these integrals that all 10 steps in the final TayInt algorithm were necessary for

I246 and I59 at all orders in ϵ.



(a) (b)

(c)

Figure 13.30: I59 below threshold, calculated with four orders and three partitions at; (a) $\mathcal{O}(\epsilon^0)$, (b) $\mathcal{O}(\epsilon^1)$, (c) $\mathcal{O}(\epsilon^2)$ with $u = -59858$ GeV^2, $m_2 = \frac{1}{\sqrt{2}} m_1$ and $m_1 = 173$ GeV. The lower plots show the relative uncertainties of the TayInt approximations and numerical SecDec results, respectively.

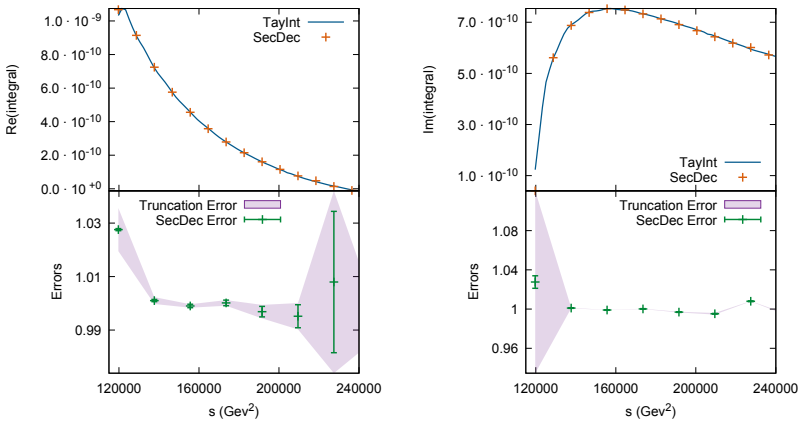


Figure 13.31: The I59 Integral calculated at ϵ^0 with with a fourth-order Taylor expansion for which high-uniform, low-uniform and varied partition sets were used for each subsector. The scale s is over the $4m_1^2$ threshold, with $u = -59858$ GeV^2, $m_2 = \frac{1}{\sqrt{2}}m_1$ and $m_1 = 173$ GeV. The lower plots show the relative TayInt and SecDec uncertainties, respectively.

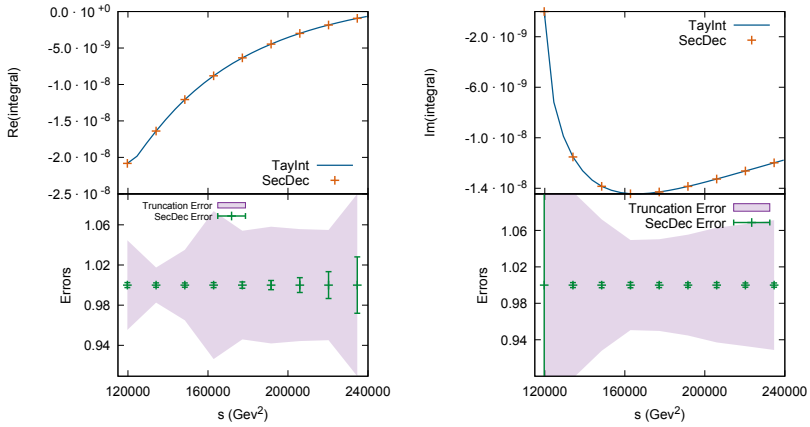


Figure 13.32: The I59 Integral calculated at ϵ^1 with a fourth-order Taylor expansion for which high-uniform, low-uniform and varied partition sets were used for each subsector. The scale s is over the $4m_1^2$ threshold, with $u = -59858$ GeV^2, $m_2 = \frac{1}{\sqrt{2}}m_1$ and $m_1 = 173$ GeV. The lower plots show the relative TayInt and SecDec uncertainties, respectively.

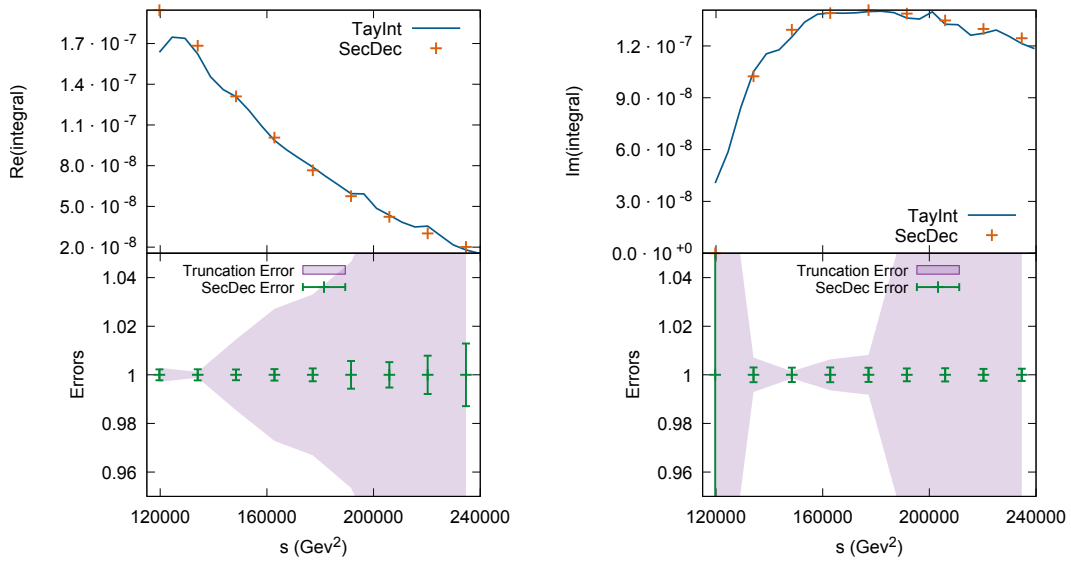


Figure 13.33: The I59 Integral calculated at ϵ^2 with a fourth-order Taylor expansion for which high-uniform, low-uniform and varied partition sets were used for each subsector. The scale s is over the $4m_1^2$ threshold, with $u = -59858$ GeV^2, $m_2 = \frac{1}{\sqrt{2}}m_1$ and $m_1 = 173$ GeV. The lower plots show the relative TayInt and SecDec uncertainties, respectively.

To reinforce how the objectives O1-3 are satisfied for maximally complicated two-loop Feynman integrals, the mean difference between the TayInt approximations and SecDec results for I246 and I59 up to order ϵ^2 over their threshold, Δ, is tabulated in Table 13.3. All TayInt approximations are based on a fourth-order, variably partitioned (as determined by the final TayInt algorithm), Taylor expansion. The mean difference between the TayInt approximations and the analytic results for I503 (over orders ϵ^{-4} to ϵ^{-1}) is also shown to illustrate how TayInt can be successfully applied to divergent two-loop Feynman integrals. The algebraic approximations for I246 are 578.9 MB in size while for I59 they total 479.6 MB. For the divergent I503, which has both considerably more and larger subsectors, they occupy 3.24 GB. The greater relative deviation observed for the Feynman integrals shown in Table 13.3 compared to those in Table 13.2 is due to their greater complexity. In particular, the Feynman integrals I246 and I59 are considerably more difficult to approximate over threshold using a Taylor expansion than I10 and I39. Therefore, in the case of the latter the accuracy is already guaranteed by the contour configurations chosen by TayInt and a high-uniform partitioning can be used to arbitrarily increase the precision of the approximation. However, in the case of the former the contour choice and the partition set are needed to ensure the accuracy of the approximation, and the partitioning cannot be raised arbitrarily to increase the precision

Table 13.3: The mean difference Δ (over epsilon orders ϵ^0-ϵ^2 for I246 and I59 and ϵ^{-4}-ϵ^{-1} for I503) between TayInt and SecDec, normalised to the SecDec result. The kinematic points are $u \in [4m_1^2, 16m_1^2] = [119716, 478864]$ GeV^2 for I246 and I59 and $s \in [4m_1^2, -m_\mu^2] = [0, -11166.6]$ MeV^2 for I503.

Graph	Algorithm Steps Used	Re (Δ)	Im (Δ)
I503	U1-BT2	0.00329	N/A
I246	U1-OT10	0.0340	0.0237
I59	U1-OT10	0.0262	0.0109

(as this would decrease the accuracy of the approximation for some subsectors).

13.4 Parallelisation

An additional feature offered by the TayInt program is the ability to parallelise the calculation of the subsectors of a Feynman integral by editing the file `NAMECALCrun` so that `Y` is replaced by the desired number of concurrent subprocesses. The effect of such a parallelisation is illustrated in Table 13.4.

Table 13.4: The impact of parallelising (`Y=4`) on the runtime of the entire TayInt program, for I503, which has 26, 100, 162 and 200 subsectors at the order $\epsilon^{-4}, \epsilon^{-3}, \epsilon^{-2}, \epsilon^{-1}$ respectively and I10, which has eight subsectors for finite ϵ.

TayInt Algebraic Run Time (s)		
Integral	Parallelised	
Name	No	Yes
I503 ϵ^{-4}	132	81
I503 ϵ^{-3}	903000	31020
I503 ϵ^{-2}	1540900	102060
I503 ϵ^{-1}	1900800	211200
I10 ϵ^0	5505	2455
I10 ϵ^1	6856	5142
I10 ϵ^2	16571	13479

13.5 Summary

With the discussion of parallelisation, the description of the TayInt program concludes. It has been shown that the objectives O1-3 of the final TayInt algorithm have been

achieved up to the highest level of complexity currently considered for Feynman integrals: two-loop, four-point and elliptic, verifying that the principles of the final TayInt algorithm are valid. Various different aspects of the final TayInt algorithm are needed to achieve these objectives for different Feynman integrals. Moreover, the promotion of the final algorithm to an automated and parallelisable program in Python indicates that TayInt is ready for application to phenomenological calculations. The next step is to calculate algebraic approximations for a complete integral family at the two-loop level, before moving on to a full phenomenological process.

14 | Conclusion

TayInt is a new program which automates an algorithm that calculates systematic approximations to Feynman integrals algebraically in the kinematic invariants whose numerical counterparts have validity in all kinematic regions and can be made arbitrarily precise.

The program takes the propagators as input and works with subsector integrals generated by the internal use of version three of the program SecDec within the TayInt code. The actual integration is facilitated via a Taylor expansion in the integration parameters. The accuracy is bolstered by conformal mappings and partitioning of the integrand before performing the Taylor expansion. The validity over threshold is ensured by performing an algorithmically determined variable transformation and partitioning, which implements the correct analytical continuation of the integrand into the complex plane. Systematic approximations can then be obtained to higher orders in the dimensional regulator ϵ, both above and below mass thresholds.

The application of TayInt was demonstrated using two-loop three-point and four-point Feynman integrals with an internal mass including the maximally difficult non-planar and elliptic cases, which were also used to illustrate several features and virtues of the final TayInt algorithm.

The TayInt program expands upon the SecDec framework and is in principle applicable to Feynman integrals with an arbitrary number of loops and any number of kinematic scales. Its practical application is ensured by the parallelisation of the calculation for distinct subsectors, the proven ability to tackle divergent integrals, and the use of partitioning rather than higher orders to control the accuracy and precision of the Taylor expansion.

Based on the difficulty and variety of the applications considered in this thesis, there is every reason to believe that the TayInt program can be applied to many two-loop and three-loop problems of high phenomenological interest, where closed analytical expressions cannot be obtained, in a fully-automated manner.

A | The TayInt Threshold-Finding Algorithm Glossary

In this Appendix a glossary of all the terms relevant to the threshold-finding algorithm is given, step by step.

A.1 Step T1

1. `SecList`: a list containing the subsectors of the Feynman integral to be calculated.
2. `kineinv`: a list containing the kinematic scales in the Feynman integral to be calculated.
3. `FeynList`: the list of Feynman parameters t_j appearing in the subsectors of the Feynman integral to be calculated.
4. `KineMassesVal`: a list containing the numerical values of the distinct masses contained by the Feynman integral to be calculated.
5. `kinemassNZ`: a list containing the non-trivial symbolic mass scales in the Feynman integral to be calculated.
6. `kinemassNZCombos`: the nested list of all the possible subsets of `kinemassNZCombos`.
7. `ThreshCand`: a list of all the possible values at which the propagators of the Feynman integral could potentially go on-shell based on their kinematic structure, from which the thresholds will be taken.
8. `ScanList`: a nested list of dimension `{49,Length[kineinv]}` which contains the numerical values for each kinematic scale of the Feynman integral used to numerically evaluate the ratios of the orders in the subsequent Taylor expansion used to choose the thresholds from `ThreshCand`. In `ScanList`, all of the kinematic scales (each slightly offset) are varied.
9. `ScanListb`: a nested list of dimension `{49,Length[kineinv]}` which contains the numerical values for each kinematic scale of the Feynman integral used to numerically evaluate the ratios of the orders in the subsequent Taylor expansion

used to choose the thresholds from `ThreshCand`. In `ScanListb`, only the kinematic scales the user wants varied are allowed to run.

A.2 Step T2

1. `TayList0-TayList6`: these are lists of length `Length[ScanList]` containing integrated results of each order, from zero to six in the Taylor expansion of the full Feynman integral evaluated at each kinematic set in `ScanList`. The corresponding lists `TayList0-TayList6` are those evaluated at the kinematic sets of values given in `ScanListb`.

2. `TayList20-TayList64`: these are the lists of the ratios between the integrated orders in the Taylor expansion of the Feynman integral indicated by the last two numerals in the name. For example, `TayList20` is obtained by dividing the absolute value of `TayList2` and `TayList0`. Consequently, these lists are also of length `Length[ScanList`. The corresponding lists of ratios `TayList20b-TayList64b` are those evaluated at the kinematic sets of values given in `ScanListb`.

3. `TayList` is a nested list of length `7`, containing as sub-lists all of the ratio lists `TayList20-TayList64`. The corresponding nested list of ratios `TayListb` are those evaluated at the kinematic sets of values given in `ScanListb`.

A.3 Step T3

1. `BunchPos`: is a nested list with outer dimension `Length[TayList]` within which each sub-list can have a different dimension, equal to the number of kinematic points in `ScanList` at which sustained breakdown of the corresponding order of the Taylor expansion occurs. The sub-lists contain the integers denoting the position in `ScanList` of the sustained breakdown for each order ratio. The corresponding nested list of integers `BunchPosb` correspond to the positions of sustained breakdown in `ScanListb`.

2. `ThreshBunches`: is a nested list with outer dimension `Length[TayList]`, second dimension equal to `Length[BunchPos[[i]]` where i runs from 1 to 7, and inner dimensions corresponding to the number of consecutive kinematic points at which sustained breakdown occurs in each order ratio. It is obtained by grouping each set of consecutive integers in the sub-lists in `BunchPos` into lists of their own. The corresponding nested list of integers `ThreshPosb` contains the grouped positions of consecutive sustained breakdown in `ScanListb`.

A.4 Step T4

1. `ThreshFac`: a nested list of dimensions equal to those of `ThreshBunches}`. It contains the numerical values of the kinematic points found at the positions given in

`ThreshBunches`. The corresponding nested list of rational numbers in `ThreshFacb` contains the grouped kinematic values of consecutive sustained breakdown in `ScanListb`.

A.5 Step T5

1. `ThreshFacDispl`: a nested list of dimensions `{Length[TayList],Length[ThreshCand]}`. Each sub-list contains the minimum difference between each potential threshold in `ThreshCand` and the positions of sustained breakdown in `ThreshFac`. The corresponding nested list of rational numbers in `ThreshFacDisplb` contains the differences between the potential thresholds and the kinematic values at which sustained breakdown occurred in `TayListb`.

2. `ThreshPos`: a nested list of dimensions

   ```
   {Length[TayList],Length[Cases[ThreshFacDispl[[i]],
   _?(# < 0.5*Total[KineMassesVal^2] &)]},
   ```
   (A.1)

   where `i` labels runs from `1` to `Length[TayList]`. Each sub-list contains the entries of `ThreshCand` selected using `ThreshFacDispl`. The corresponding nested list of rational numbers in `ThreshFacDisplb` contains the entries of `ThreshCand` selected using `ThreshFacDisplb`.

3. `ThreshPosFin`: This is a list of the union of all the sub-lists in `ThreshPos` and `ThreshPosb` that contains the thresholds as determined by TayInt.

B | Final TayInt Algorithm

B.1 The TayInt Calculation Code

To actually calculate results for the representation of a generic Feynman integral obtained by the TayInt algorithm, three steps are necessary. The first, performed by `COMPDiscTaySeriesIntegrator`, is to produce an integrated Taylor expansion for each subsector of the Feynman integral that is algebraic in the expansion points and limits of integration. In this form, the contour configurations and partition sets determined by the TayInt algorithm can then be used to produce a result for the Feynman integral that is algebraic in the kinematic scales and is primed to produce accurate and precise numerical results at any kinematic point. These results are produced used `COMPDiscTaySliceOutput`, by inserting the relevant limits of integration, whose direction is determined by the contour configuration and whose size determined by the partition set prescribed by the algorithm. The midpoints of these algorithmically determined integration regions are inserted as the expansion points. Finally, arbitrary kinematic points can be inserted into these algebraic results to produce precise and accurate numerical results for the starting Feynman integral. The computation of the algebraic results is described in detail below, referring to I59 as an example of the working of the TayInt code.

B.1.1 Algebraic Calculation Code

Algebraic Taylor Expansion and Integration

1. This is performed by the Mathematica module

   ```
   COMPDiscTaySeriesIntegrator[func,varlist_?VectorQ,
   conflist_?VectorQ,startord_,endord,name,sec]
   ```

   which takes the arguments:

   a) `func`: this is the subsector, for example, the first of I59 at order ϵ^1, given below in Eq. (B.1), with the Feynman parameters transformed to the complex space:

$$
\begin{aligned}
\mathrm{I59}^{1}_{01} = {} & (\log[t_4] - 3\log[1 + t_0 + t_1 + t_2 + t_3 + (1 + t_0)\cdot(t_1 + t_2 + t_3)t_4] \\
& 2\log[-s\cdot t_3\cdot(1 + t_1 + t_1t_4) - t_0(u\cdot t_2 + s\cdot t_1t_3t_4) \\
& -u\cdot t_0\cdot(1 + t_1 + (t_1 + t_2 + t_3)\cdot t_4) \\
& +m^2\cdot(1 + t_0 + t_1 + t_2 + t_3)\cdot(1 + t_0 + t_1 + t_2 + t_3 + (1 + t_0)\cdot(t_1 + t_2 + t_3)\cdot t4)]\Big) \\
& (u\cdot t_0t_2 + s\cdot t_3 + s\cdot t_1t_3 + s\cdot(1 + t_0)t_1t_3t_4 \\
& +m_h^2\cdot t_0(1 + t_1 + (t_1 + t_2 + t_3)\cdot t_4) \\
& -m^2\cdot(1 + t_0 + t_1 + t_2 + t_3)(1 + t_0 + t_1 + t_2 + t_3 + (1 + t_0)\cdot(t_1 + t_2 + t_3)\cdot t_4)\Big)^{-2} \ .
\end{aligned}
\tag{B.1}
$$

b) `varlist`: the list of Feynman parameters, which, after performing the complex transformation [1], reads {`theta[0],theta[1],theta[2],theta[3],theta[4]`} in the case of I59.

c) `conflist`: the list of conformally mapped Feynman parameters, which is still {`theta[0],theta[1],theta[2],theta[3],theta[4]`} in the case of I59 as conformal mappings do not improve the convergence when the original damage is due to threshold discontinuities, as these are tied to the region of integration.

d) `startord`: the order at which the Taylor expansion will begin, which, in the case of I59 is zero.

e) `endord`: the order at which the Taylor expansion will end, which, in the case of I59 is order four. The order can be so low because the convergence will be improved by the TayInt partitioning method [1].

f) `name`: the name chosen by the user to identify this result, for example `I59Eps0`.

g) `sec`: the subsector number, for example, `1`.

Before outlining the tasks performed by this module, a general comment on its methodology is required. The integral of a Taylor expansion,

$$
\int_{i_l}^{i_h} dx \left(f(e) + (x-e)f^{(1)}(e) + \frac{(x-e)^2}{2!}f^{(2)}(e)... + \frac{(x-e)^{n-1}}{(n-1)!}f^{(n-1)}(e) \right) , \tag{B.2}
$$

with limits of integration $[i_l, i_h]$ and an expansion point e can be rewritten as:

$$
\int_{i_l+e}^{i_h+e} dx \left(f(e) + xf^{(1)}(e) + \frac{x^2}{2!}f^{(2)}(e)... + \frac{x^{n-1}}{(n-1)!}f^{(n-1)}(e) \right) , \tag{B.3}
$$

via the transformation $x' = x + e$ followed by the re-labelling $x = x'$. The TayInt code always calculates integrated Taylor expansions using the latter form, B.3, because it is computationally simpler and thus the integration is significantly faster.

For the purpose of explanation the module itself can be split into three parts, the setup lists, the algebraic computation and the output.

Setup Lists

The setup lists are displayed and described below, accompanied by the output (grey) in the case of I59 subsector one at order ϵ^1 (which, after the complex transformation occupies 66 lines in Mathematica after running the `Simplify[]` command). First, the mapping is performed (if no mapping is desired then the same list should be entered for both `varlist` and `conflist`), returning `confFunc`, the mapped function multiplied by the relevant Jacobian. Once this is done, the first setup list is that containing variables raised to an arbitrary power, from which subsets will be taken and multiplied out to construct the algebraic part of the Taylor expansion. This list is termed `powlist` and is created as follows:

```
powlist=pow[#]&/@Range[1,Length[varlist]]                    (B.4)
{pow[1],pow[2],pow[3],pow[4],pow[5]}
powvarlist = varlist[[#]]^(powlist[[#]])
&/@Range[1,Length[varlist]]                                  (B.5)
{theta[0]^pow[1], theta[1]^pow[2], theta[2]^pow[3],
theta[3]^pow[4], theta[4]^pow[5]} .
```

Next, the lists of expansion parameters (`elist`), limits of integration (`illist`, `ihlist`) and the domains of integration (`intlimlist`), which will be mapped over to produce the results for the integrated Taylor expansion, are created:

```
elist=e[#]&/@Range[1,Length[varlist]]                        (B.6)
{e[1],e[2],e[3],e[4],e[5]}
illist=il[#]&/@Range[1,Length[varlist]]                      (B.7)
{il[1],il[2],il[3],il[4],il[5]}
ihlist=ih[#]&/@Range[1,Length[varlist]]                      (B.8)
{ih[1],ih[2],ih[3],ih[4],ih[5]}
intlimlist=(varlist[[#]],illist[[#]],ihlist[[#]])
&/@Range[1,Length[varlist]]                                  (B.9)
{{theta[0],il[1],ih[1]}, {theta[1],il[2],ih[2]},
{theta[2],il[3],ih[3]}, {theta[3],il[4],ih[4]},
{theta[4],il[5],ih[5]}} .
```

Taylor Expansion and Integration

In order to construct the Taylor expansion of the input function as quickly as possible, it is necessary to be able to produce the derivative terms in an efficient manner. The fastest way to perform differentiation in Mathematica is to map the derivative operator over lists which specify the order to which the partial derivative in each variable is computed. First, the list of all combinations of derivative orders in each variable that contribute to the order of the Taylor expansion requested, `sum`, is generated:

```
sum=((Flatten[Table[If[startord<=(Plus@@sumindexlist)<=endord,
sumindexlist,alpha],Evaluate[Sequence@@sumlimlist]],
Length[varlist]-1])//.alpha->Sequence[])                    (B.10)
{{0,0,0,0,0},{0,0,0,0,1},...
...{3,1,0,0,0},{4,0,0,0,0}},                                (B.11)
```

to which the relevant variable is then appended, yielding:

```
Table[derlist[i]=({varlist[[#]],sum[[i]][[#]]})
&/@Range[1,Length[varlist]],{i,1,Length[sum]}]              (B.12)
{{{theta[0],0},{theta[1],0},{theta[2],0},
{theta[3],0},{theta[4],0}},...{{theta[0],4},{theta[1],0},
{theta[2],0},{theta[3],0},{theta[4],0}}} .
```

The derivative of `confFunc` is then mapped over the list of orders specified in Eq. B.12, the necessary factorial terms are included and the variables set to the expansion points given by B.6. This generates a list of all the Taylor coefficients that contribute to the Taylor series of `func` at the order specified by `startord` and `endord`. The piece of code that performs this reads:

```
(TaylorFunc[#]=1/(Times@@(sum[[#]]!))*(D[confFunc,
Evaluate[Sequence@@derlist[#]]]/.Thread[varlist->elist]
/.Thread[N[varlist]->elist]))&/@Range[1,Length[sum]] .       (B.13)
```

At this point, the result could be generated by multiplying together the `TaylorFunc[#]` and relevant `powvarlist` elements and integrating. However, the speed of the code is an objective of paramount importance and no opportunity to increase it should be allowed to pass by. Therefore, as it is not necessary to integrate over the entries in `TaylorFunc`, because they contain no integration variables, only the relevant

products of the `powvarlist` elements are integrated over. Because of the complexity of the subsectors of two-loop Feynman integrands, this separation reduces the run time of this part of the code by a factor of 3160, or a reduction of 99.968%. The minimal necessary integration is performed as follows:

```
IntFunc=Integrate[Apply[Times,((powvarlist
/.Thread[powlist->sum[[#]]])&/@Range[1,Length[sum]]),
1],Evaluate[Sequence@@intlimlist]] ,                    (B.14)
```

after which it only remains to multiply the two lists together element-wise to yield each piece (labelled by `#`) in the integrated Taylor expansion for the input complex-mapped subsector:

```
(TaylorResult[#]=((TaylorFunc[#]*IntFunc[[#]])))
&/@Range[1,Length[sum]]!.                               (B.15)
```

These pieces are then stored as `.txt` files to give a piecewise expression for the integrated Taylor expansion, algebraic in the expansion points and limits of integration, which will be recycled to generate results in partitions of the integrand. Note that given the relatively small number of pieces at this stage, it would be faster to sum the pieces and output them as one result, rather than storing each piece, with the former taking place a factor of 2124.49 faster. However, in the next step the expansion points and limits of integration are inserted to generate the results in each partition of the integrand and inserting values on a piecewise basis is a factor of 9000 faster than replacing values in the entire file. Moreover, the insertion step is slower and so each factor of speed increase there translates into more of a gain in real time. Additionally, when the kinematic parameters are replaced in the final step, it is twelve times faster if the algebraic results in each partition have been simplified. Using the `Simplify` operation on a piecewise basis does not alter the computation time necessary for the second step but runs indefinitely (due to the complexity of the integrands being considered) if the entire integrated Taylor expansion is used. In conclusion, there are three features of the `COMPDiscTaySeriesIntegrator` module that are crucial to the efficiency of the TayInt code:

- **The use of `Map`:** all iterated operations are performed using the `Map` operator applied to lists (known as vectorisation), which is much faster than using loops or the `Table` command in Mathmatica.
- **Minimising the argument of `Integrate`.**The most computationally intensive tasks are performed as little as is mathematically possible. The integration step is the most time consuming part of the `COMPDiscTaySeriesIntegrator` module. Therefore, only the parts of the integrand that must be integrated are passed to the `Integrate` function and they are simplified as much as possible.

- **Sacrificing efficiency now for greater gains later.** It is faster to output the algebraic Taylor integrated result all at once rather than splitting it up, because this minimises the number of times files need to be written. However, when the expansion points and limits of integration are inserted in the next step to generate the algebraic results in each partition of the integrand, it is much faster to proceed on a piece-by-piece basis, minimising the sizes of the files that need to be processed by the insertion operation. Hence each term in the integrated Taylor expansion is stored in a separate file.

Partitioning

2. The terms of the algebraic integrated Taylor expansion are then used to generate results for the integrated Taylor expansion of the subsector in different parts of its hypersurface of integration. These partitioned results are algebraic in the kinematic scales and are produced using the Module:

   `COMPDiscTaySliceOutput[sec_,name_,varlist,conflist_,startord_, endord_,limintlist_,ralist] ,`

   which has the arguments `sec`, `name`, `varlist`, `conflist`, `startord`, `endord` in common with `COMPDiscTaySeriesIntegrator` and the following additional arguments:

   a) `limintlist`: the list of integration boundaries which specify the contour configuration, obtained in `CCFinal`, that is used for integrating each subsector. In the case of I59 subsector one this reads

   $$\texttt{limintlist} = \{\{0,-\pi\},\{0,\pi\},\{0,-\pi\},\{0,-\pi\},\{0,-\pi\}\}\ . \tag{B.16}$$

   b) `ralist`: the number of partitions per integration variable, which reads:

   $$\begin{aligned}\texttt{ralist} =&\{\{1,1,1,1,1,1,1,1\},\\ &\{1,1,1,1,1,1,1,1\},\\ &\{1,1,1,1,1,1,1,1\},\\ &\{1,1,1,1,1,1,1,1\},\\ &\{1,1,1,1,1,1,1,1\ \}\},\end{aligned} \tag{B.17}$$

   in the case of I59.

   In order to input the terms of the integrated Taylor expansion, the list of terms, indexed according to the orders of the derivatives, must again be created. However, as the code that performs this in `COMPDiscTaySliceOutput` is identical to that contained in `COMPDiscTaySeriesIntegrator` and elaborated upon in Eq. (B.11), it will not be discussed again here. The expansion points and boundaries of integration are then inserted into the terms of this algebraic integrated Taylor expansion to create the results in each partition of the integrand. To achieve this, a list must be created which will allow the code to map each algebraic term over the relevant numbers describing each partition.

The partition lists

To generate this list, a zero is first appended to the list of partitions for each variable (`ralist`, Eq. B.17), with the code:

```
(ratlist[#]=Append[ralist[[#]],0])&/@Range[1,Length[varlist]] ,
```
(B.18)

which yields

```
ratlist={{1,1,1,1,1,1,1,1,0},
        {1,1,1,1,1,1,1,1,0},
        {1,1,1,1,1,1,1,1,0},
        {1,1,1,1,1,1,1,1,0},
        {1,1,1,1,1,1,1,1,0}} ,
```
(B.19)

for I59. The lengths of each partition are then created by mapping the `limintlist` over the number and relative sizes of the partitions described by `ratlist`, using the code:

```
(delta[#]=Total[ratlist[#]])&/@Range[1,Length[varlist]]
```
(B.20)

```
                {6,6,6,6,6}
(difflist[#]=Table[ratlist[#][[i]]/delta[#]*intstoplist[[#]],
                {i,1,Length[ratlist[#]]}])
                &/@Range[1,Length[varlist]]
                {{-π/6,-π/6,-π/6,-π/6,-π/6,-π/6, 0},
                {{ π/6, π/6, π/6, π/6, π/6, π/6, 0},...
                ...,{{-π/6,-π/6,-π/6,-π/6,-π/6,-π/6, 0} ,
```
(B.21)

where the output in grey shows the result in the case of I59, for which

$$\texttt{limintlist}=\{\{0,-\pi\},\{0,\pi\},\{0,-\pi\},\{0,-\pi\},\{0,-\pi\}\}\ .$$

With this information to hand, the boundaries of integration of each partition can finally be constructed, for which the code:

```
(deltalist[#]=Table[Sum[difflist[#][[i]],{i,1,j}],
                {j,1,Length[difflist[#]]-1}])
                &/@ Range[1,Length[varlist]]
                &/@ Range[1,Length[varlist]]                    (B.22)
```

$$\{\{-\frac{\pi}{6},-\frac{\pi}{3},-\frac{\pi}{2},-\frac{2\pi}{3},-\frac{5\pi}{6},-\pi,\ 0\},\ldots$$
$$\{\{\ \frac{\pi}{6},\ \frac{\pi}{3},\ \frac{\pi}{2},\ \frac{2\pi}{3},\ \frac{5\pi}{6},\ \pi,\ 0\},\ldots$$
$$\{\{-\frac{\pi}{6},-\frac{\pi}{3},-\frac{\pi}{2},-\frac{2\pi}{3},-\frac{5\pi}{6},-\pi,\ 0\},\ldots$$

```
(intlist[#]=Partition[Prepend[deltalist[#],intstartlist[[#]]],2,1])
                &/@Range[1,Length[varlist]]
                &/@Range[1,Length[varlist]]                     (B.23)
```

$$\{\{\{0,-\frac{\pi}{6}\},\{-\frac{\pi}{6},-\frac{\pi}{3}\},\{-\frac{\pi}{3},-\frac{\pi}{2}\},\{-\frac{\pi}{2},-\frac{2\pi}{3}\},$$
$$\{-\frac{2\pi}{3},-\frac{5\pi}{6}\},\{-\frac{5\pi}{6},-\pi\}\},\ldots$$
$$\{\{\{0,\ \frac{\pi}{6}\},\{\ \frac{\pi}{6},\ \frac{\pi}{3}\},\{\ \frac{\pi}{3},\ \frac{\pi}{2}\},\{\ \frac{\pi}{2},\ \frac{2\pi}{3}\},$$
$$\{\ \frac{2\pi}{3},\ \frac{5\pi}{6}\},\{\ \frac{5\pi}{6},\ \pi\}\},\ldots$$
$$\{\{\{0,-\frac{\pi}{6}\},\{-\frac{\pi}{6},-\frac{\pi}{3}\},\{-\frac{\pi}{3},-\frac{\pi}{2}\},\{-\frac{\pi}{2},-\frac{2\pi}{3}\},$$
$$\{-\frac{2\pi}{3},-\frac{5\pi}{6}\},\{-\frac{5\pi}{6},-\pi\}\},$$

is utilised. Once again, the output in grey shows the result in the case of I59. The expansion points of each partition are simply the midpoints of the end points of integration, so are readily obtained from the above list. The remainder of the `COMPDiscTaySliceOutput` Module uses these lists to map the algebraic results obtained from `COMPDiscTaySeriesIntegrator` to the result of the integral in each partition.

B.1.2 Numerical Calculation Code

Once the Taylor expansion and integration is carried out, what is left is a result for the Feynman integral in terms of its kinematic scales, which will converge to the correct result even when over-threshold values are inserted for these scales. This has been guaranteed by the choice of contour along which the integration was performed and is computationally performed by the function `COMPKine`.

B.1.3 Final TayInt Algorithm Glossary

OT1

1. `SecList`: a list containing the subsectors of the Feynman integral to be calculated.

2. `kineinv`: a list containing the kinematic scales in the Feynman integral to be calculated.

3. `FeynList`: the list of Feynman parameters t_j appearing in the subsectors of the Feynman integral to be calculated.

4. `CCNIELcalc`: a nested list of dimensions $\{2^{\texttt{Length[FeynList]}}$, `Length[FeynList], 2}` . Each potential configuration of each integration variable is listed.

5. `CCNIELpartialcalc`: a nested list containing four layers. The first is the number of partial complex transformations. The second is the number of possible contour configurations within that transformation. The third is the list of orientations for each integration variable on that contour, the fourth is the list of the start andend points of integration with that orientation imposed.

6. `CAdomsec`: a list containing the subsectors of the Feynman integral to be calculated after a full complex mapping has been performed.

7. `CAdomsecSub`: a nested list of dimension `{Length[SecList], Length[CCNIELpartialcalc] }` containing the subsectors of the Feynman integral to be calculated after each possible partial complex mapping has been performed.

8. `varlist`: the list of Feynman parameters after a complex mapping θ_j in the complex-mapped subsectors of the Feynman integral to be calculated.

9. `TrainKine`: a nested list of dimensions `{11, Length[kineinv]}` where `kineinv` is the list of kinematic scales in the Feynman integral. It contains eleven lists of numerical values for these scales, over the range from the first threshold bounding from below the kinematic points entered by the user up to ten times the lowest threshold of the integral.

10. `CrossVal1Kine`: a nested list of dimensions `{11, Length[kineinv]}` where `kineinv` is the list of kinematic scales in the Feynman integral. It contains eleven lists of numerical values for these scales, from the first 20% of the range of `TrainKine`.

11. `CrossVal2Kine`: a nested list of dimensions `{11, Length[kineinv]}` where `kineinv` is the list of kinematic scales in the Feynman integral. It contains eleven lists of numerical values for these scales, between 0.35 and 0.55 times the highest point in `TrainKine`.

12. `CrossVal3Kine`: a nested list of dimensions `{11, Length[kineinv]}` where `kineinv` is the list of kinematic scales in the Feynman integral. It contains eleven lists of numerical values for these scales, between 0.7 and 1.0 times the highest point in `TrainKine`.

13. `NoPart`: a nested list of dimensions `{Length[FeynList], 1}` which contains the plain partition sets for each integration variable in the complex subsectors. It is used to generate results for the subsectors without any partitioning, in order to quantify the effect that adding a partitioning has on a given contour and hence assess the suitability of the potential contour representations of each subsector.

14. `TestPart`: a nested list of dimensions `{Length[FeynList], Length[FeynList]}`, which contains a list of partition sets for each integration variable, one of which is `{1,1,1}`, with the remainder being `{1}`, in each case. This nested list is used to generate results for the subsectors of the Feynman integral on each potential full or partial contour configuration to determine how each of their integration variables responds to being partitioned. This information allows the potential contour configurations to be classified according to how many of the integration variables respond in a certain way and hence ranked. Based on this ranking, the optimal full and partial contours can be selected for each subsector of the Feynman integral to be calculated.

OT2

1. `TrainRatSensFracSec[k]`: for each subsector `k`, this is a nested list of dimensions `{Length[CCNIELcalc], Length[FeynList]}`. It contains the ratios of the results generated using `TestPart` and `NoPart`, on each full contour configuration, for each variable of integration, with the kinematic scales replaced by the entries in `TrainKine`, over which the mean absolute value is taken.
`TrainReRatSensFracSec[k]` is the list of real parts of each entry in `TrainRatSensFracSec[k]` and `TrainImRatSensFracSec[k]` is the list of imaginary parts of each entry in `TrainRatSensFracSec[k]`.

2. `Cross1RatSensFracSec[k]`: for each subsector `k`, this is a nested list of dimensions `{Length[CCNIELcalc], Length[FeynList]}`. It contains the ratios of the results generated using `TestPart` and `NoPart`, on each full contour configuration, for each variable of integration, with the kinematic scales replaced by the entries in `CrossVal1Kine`, over which the mean absolute value is taken.
`Cross1ReRatSensFracSec[k]` is the list of real parts of each entry in `Cross1RatSensFracSec[k]` and `Cross1ImRatSensFracSec[k]` is the list of imaginary parts of each entry in `Cross1RatSensFracSec[k]`.

3. `Cross2RatSensFracSec[k]`: for each subsector `k`, this is a nested list of dimensions `{Length[CCNIELcalc], Length[FeynList]}`. It contains the ratios of the results generated using `TestPart` and `NoPart`, on each full contour configuration,

for each variable of integration, with the kinematic scales replaced by the entries in `CrossVal2Kine`, over which the mean absolute value is taken.
`Cross2ReRatSensFracSec[k]` is the list of real parts of each entry in `Cross2RatSensFracSec[k]` and `Cross2ImRatSensFracSec[k]` is the list of imaginary parts of each entry in `Cross2RatSensFracSec[k]`.

4. `Cross3RatSensFracSec[k]`: for each subsector `k`, this is a nested list of dimensions `{Length[CCNIELcalc], Length[FeynList]}`. It contains the ratios of the results generated using `TestPart` and `NoPart`, on each full contour configuration, for each variable of integration, with the kinematic scales replaced by the entries in `CrossVal3Kine`, over which the mean absolute value is taken.
`Cross3ReRatSensFracSec[k]` is the list of real parts of each entry in `Cross3RatSensFracSec[k]` and `Cross3ImRatSensFracSec[k]` is the list of imaginary parts of each entry in `Cross3RatSensFracSec[k]`.

5. `TrainPartialRatSensFracSec[k]`: for each subsector `k`, this is a nested list of dimensions `{Dimensions[CCNIELpartialcalc], Length[FeynList]}`. It contains the ratios of the results generated using `TestPart` and `NoPart`, for each possible partial complex mapping, on each possible partial contour configuration within each mapping, for each variable of integration, with the kinematic scales replaced by the entries in `TrainKine`, over which the mean absolute value is taken.

   `TrainRePartialRatSensFracSec[k]` is the list of real parts of each entry in `TrainPartialRatSensFracSec[k]` and `TrainImPartialRatSensFracSec[k]` is the list of imaginary parts of each entry in `TrainPartialRatSensFracSec[k]`.

6. `Cross1PartialRatSensFracSec[k]`: A list of dimensions `{Dimensions[CCNIELpartialcalc], Length[FeynList]}}`. It contains the ratios of the results generated using `TestPart` and `NoPart`, for each possible partial complex mapping, on each possible partial contour configuration within each mapping, for each variable of integration, with the kinematic scales replaced by the entries in `CrossVal1Kine`, over which the mean absolute value is taken.
`Cross1RePartialRatSensFracSec[k]` is the list of real parts of each entry in `Cross1PartialRatSensFracSec[k]` and `Cross1ImPartialRatSensFracSec[k]` is the list of imaginary parts of each entry in `Cross1PartialRatSensFracSec[k]`.

7. `Cross2PartialRatSensFracSec[k]`: for each subsector `k`, this is a nested list of dimensions `{Length[CCNIELcalc],Length[FeynList]}`. It contains the ratios of the results generated using `TestPart` and `NoPart`, for each possible partial complex mapping, on each possible partial contour configuration within each mapping, for each variable of integration, with the kinematic scales replaced by the entries in `CrossVal2Kine`, over which the mean absolute value is taken.

   `Cross2RePartialRatSensFracSec[k]` is the list of real parts of each entry in `Cross2PartialRatSensFracSec[k]` and

`Cross2ImPartialPartialRatSensFracSec[k]` is the list of imaginary parts of each entry in `Cross2PartialRatSensFracSec[k]`.

8. `Cross3PartialRatSensFracSec[k]`: for each subsector `k`, this is a nested list of dimensions `{Length[CCNIELcalc],Length[FeynList]}`. It contains the ratios of the results generated using `TestPart` and `NoPart`, for each possible partial complex mapping, on each possible partial contour configuration within each mapping, for each variable of integration, with the kinematic scales replaced by the entries in `CrossVal3Kine`, over which the mean absolute value is taken.
`Cross3RePartialRatSensFracSec[k]` is the list of real parts of each entry in `Cross3PartialRatSensFracSec[k]` and `Cross3ImPartialRatSensFracSec[k]` is the list of imaginary parts of each entry in `Cross3PartialRatSensFracSec[k]`.

OT3

1. `TrainSensBadLowVarPosSec[k]`: for each subsector `k`, this list contains the positions of each partition:plain ratio in the range $[0, 0.5]$ for each full contour configuration in `CCNIELcalc`. It is of length `Length[CCNIELcalc]`.

2. `TrainSensBadHighVarPosSec[k]`: for each subsector `k`, this list contains the positions of each partition:plain ratio in the range $[1.5, \infty)$ for each full contour configuration in `CCNIELcalc`. It is of length `Length[CCNIELcalc]`.

3. `TrainLenBadLowVarPosSec[k]`: for each subsector `k` this list of length `Length[CCNIELcalc]` is a list of numbers containing the number of partition:plain ratios in the range $[0, 0.5]$ per full contour.

4. `TrainLenBadHighVarPosSec[k]`: for each subsector `k` this list of length `Length[CCNIELcalc]` is a list of numbers containing the number of partition:plain ratios in the range $[1.5, \infty]$ per full contour.

5. `TrainNullLenBadLowVarSensPosSec[k]`: for each subsector `k` this is a list of the positions of the full contours in `CCNIELcalc` with no partition:plain ratios in the range $[0, 0.5]$.

6. `TrainNullLenBadHighVarSensPosSec[k]`: for each subsector `k` this is a list of the positions of the full contours in `CCNIELcalc` with no partition:plain ratios in the range $[1.5, \infty)$.

7. `TrainNullLenBadVarSensPosSec[k]`: for each subsector `k` this is a list of the positions of the full contours in `CCNIELcalc` with no partition:plain ratios in the range $[0.5, 1.5]$.

8. `TrainMeanRatBadVarSensFracSec[k]`: for each subsector `k` this is a list of the mean partition:plain ratio for the full contours in `CCNIELcalc` with no partition:plain ratios in the range $[0.5, 1.5]$.

9. `TrainMinMeanRatBadVarSensFracSec[k]`: for each subsector `k` this is a number specifying the position of the minimum in `TrainMeanRatBadVarSensFracSec[k]`.

10. `TrainChosenBadVarRatContourSec[k]`: for each subsector `k` this is a number specifying the position of the full contour in textttCCNIELcalc which has the minimal mean of the contours in `TrainMaxLenBadVarSensPosSec[k]`.

11. `TrainSensGoodVarPosSec[k]`: for each subsector `k`, this list contains the positions of each partition:plain ratio in the range $[0.25, 1.0]$ for each full contour configuration in `CCNIELcalc`. It is of length `Length[CCNIELcalc]`.

12. `TrainLenGoodVarPosSec[k]`: for each subsector `k` this list of length `Length[CCNIELcalc]` is a list of numbers containing the number of partition:plain ratios in the range $[0.25, 1.0]$ per full contour.

13. `TrainMaxLenGoodVarSensPosSec[k]`: for each subsector `k` this is a list of the positions of the full contours in `CCNIELcalc` with the maximum number of partition:plain ratios in the range $[0.25, 1.0]$.

14. `TrainMeanRatGoodVarSensFracSec[k]`: for each subsector `k` this is a list of the mean partition:plain ratio for the full contours in `CCNIELcalc` with the maximum number of partition:plain ratios in the range $[0.25, 1.0]$.

15. `TrainMinMeanRatGoodVarSensFracSec[k]`: for each subsector `k` this is a number specifying the position of the minimum in `TrainMeanRatGoodVarSensFracSec[k]`.

16. `TrainChosenGoodVarRatContourSec[k]`: for each subsector `k` this is a number specifying the position of the full contour in textttCCNIELcalc which has the minimal mean of the contours in `TrainMaxLenGoodVarSensPosSec[k]`.

17. Each of these lists is defined in exactly the same way for the partial contours, the only difference is that they all have an extra layer corresponding to the number of partial complex mappings within which the possible contour configurations are analysed, as above.

OT4

1. `ContCrossVal1`: a nested list of dimensions `{Length[SecList], Length[CCNIELcalc]}`. It contains numbers which quantify the relative difference between the results of `Cross1RatSensFracSec[k]` and `TrainRatSensFracSec`.

2. `ContCrossVal2`: a nested list of dimensions `{Length[SecList], Length[CCNIELcalc]}`. It contains numbers which quantify the relative difference between the results of `Cross2RatSensFracSec[k]` and `TrainRatSensFracSec`.

3. `ContCrossVal3`: a nested list of dimensions `{Length[SecList], Length[CCNIELcalc]}`. It contains numbers which quantify the relative difference between the results of `Cross3RatSensFracSec[k]` and `TrainRatSensFracSec`.

4. `LowCrossVal1[k]`: for each subsector `k` this is a list specifying the positions of those full contours in `CCNIELcalc` which have an entry in `ContCrosVal1` less than 0.5.

5. `LowCrossVal2[k]`: for each subsector `k` this is a list specifying the positions of those full contours in `CCNIELcalc` which have an entry in `ContCrosVal2` less than 0.25.

6. `LowCrossVal3[k]`: for each subsector `k` this is a list specifying the positions of those full contours in `CCNIELcalc` which have an entry in `ContCrosVal3` less than 0.5.

7. `LowCrossVal[k]`: for each subsector `k` this is a list of numbers specifying the intersection of those contours in `LowCrossVal1[k]`,`LowCrossVal2[k]` and `LowCrossVal3[k]`.

8. `IntTrainCrossContSec[k]`: for each subsector `k` this is a list containing numbers which are the intersection of `rainedRatContourSec[k]` and `LowCrossVal[k]`. It quantifies whether or not the optimal full contour also generalises to other kinematic points.

OT5

1. `FuParProx`: a list of length `Length[SecList]` the entries of which numerically quantify the relative deviation between the partition:plain ratios on the optimal full and partial contour configurations, with the kinematic scales replaced by the entries of the training set, over which the mean is subsequently taken. It is used to assess whether or not the partition:plain ratios generated using the training set or the the second cross-validation set should be used to choose between the optimal full and partial contours in `ChosenContourSec[k]` and `PartialChosenContourSec[k]` for subsector `k`.

2. `FuParComp`: a nested list of dimensions `{Length[SecList], 2, Length[FeynList]}`. It concatenates the lists of absolute partition:plain ratios for the optimal full (first position, in second layer) and partial (second position in second layer) contours. `FuParReComp` is the same list but with the real part rather than the absolute value of the ratios and likewise for `FuParImComp`.

3. `NumBadVarSec[k]`: for each subsector `k` this is a bipartite list containing the number of integration variables with partition:plain less than 0.25 or greater than 1.25 for the optimal full and partial contours.

4. `BaCho[k]`: for each subsector `k` this is a number giving the position of any zeros in the `NumBadVarSec[k]` which have distinct entries. It is used to assign the first entries to `FuParDecCho` and choose the optimal full or partial contour for each subsector.

5. `NumCompVarSec[k]`: for each subsector `k` this is a bipartite list containing the number of integration variables with partition:plain less in the range $[0.8, 1.2]$ for the optimal full and partial contours.

6. `CompCho[k]`: for each subsector `k` this is a number giving the position of the maximum in the `NumCompVarSec[k]` which have distinct entries. It is used to assemble `FuParDecCho` for those subsectors not already assigned a contour using `BaCho[k]`.

7. `GoodBaDiff[k]`: for each subsector `k` this is a bipartite list containing the difference between the number of integration variables with partition:plain ratios in the range $[0.5, 0.7]$ and those in the range $[0, 0.25] \cap [1.5, \infty)$ for the optimal full and partial contours.

8. `FuParDecCho`: this is a list of numbers of length `Length[SecList]`, each entry of which is either a 1, if the optimal full contour is chosen, or a 2, if the optimal partial contour is chosen.

9. `CCFinal`: a list of length `Length[SecList]` in which each entry specifies the position in `CCNIELcalc` or `CCNIELpartialcalc` of the final chosen contour for each subsector.

OT6-8

1. `Hom3PlainAbsRat`: a list of length `Length[SecList]` which contains the mean absolute ratio of the results generated using a uniform partition set `{1,1,1}` and no partitions, for the chosen contour. The mean is taken over the kinematic training set. `Hom3PlainReRat` and `Hom3PlainImRat` are the corresponding ratios in the real and imaginary parts of the aforementioned results.

2. `Hom2PlainAbsRat`: a list of length `Length[SecList]` which contains the mean absolute ratio of the results generated using a uniform partition set `{1,1}` and no partitions, for the chosen contour, with the mean taken over the kinematic training set.

3. `Hom3Hom2AbsRat`:a list of length `Length[SecList]` which contains the mean absolute ratio of the results generated using a uniform partition set `{1,1,1}` and a uniform partition set `{1,1}`, for the chosen contour, with the mean taken over the kinematic training set.

OT9

1. `ParList[2]`: A nested list, of the form `{1,1},{1},...,{1}`, where the number of `{1}` entries is equal to `Length[FeynList]-1`, used to generate ratios for each integration variable of those subsectors for which a uniform partition set is not suitable.

2. `ParList[3]`: A nested list, of the form `{1,1,1},{1},...,{1}`, where the number of `{1}` entries is equal to `Length[FeynList]-1`, used to generate ratios for each integration variable of those subsectors for which a uniform partition set is not suitable.

3. `ParList[4]`: A nested list, of the form `{1,1,1,1},{1},...,{1}`, where the number of `{1}` entries is equal to `Length[FeynList]-1`, used to generate ratios for each integration variable of those subsectors for which a uniform partition set is not suitable.

4. `ParTest`: A nested list, of the form

```
ParTest =
{{{{1,1},{1},...,{1}},{{1},{1,1},...,{1}},{{1},{1},{1,1},...},...}
{{{1,1,1},{1},...,{1}},{{1},{1,1,1},...,{1}},{{1},{1},{1, 1,1},...},...}
{{{1,1,1,1},{1},...,{1}},{{1},{1,1,1,1},...,{1}},{{1},{1},{1,1,1,1},...},...}
```
(B.24)

   with dimension `{3,Length[FeynList], Length[FeynList]}`. It contains three lists corresponding to the use of two, three and four partitions, in each integration variable, within each of the sub-lists, with the remaining variables assigned a plain partition set in the sub-sub-lists. It is used to generate the ratios that assess the partition set suitable of each individual integration variable.

5. `PartReRat[k]`: for each subsector `k` this is a nested list of dimension `{2,Length[FeynList]}`, which contains the lists of the ratios between the results generated with; `ParTest[[2]]` and `ParTest[[1]]`, `ParTest[[3]]` and `PartTest[[2]]`. These numerically quantify the behaviour of each individual integration variable when they are calculated with a partition set.

6. `ChosenPartReSec[k]`: for each subsector `k`, this is a nestd list containing the partition sets chosen for each integration variable, to be used when calculating the real part of the Feynman integral.

7. `ChosenPartImSec[k]`: for each subsector `k`, this is a nestd list containing the partition sets chosen for each integration variable, to be used when calculating the imaginary part of the Feynman integral.

15 | Bibliography

[1] S. Borowka, T. Gehrmann, D. Hulme, Systematic approximation of multi-scale Feynman integrals, JHEP 08 (2018) 111. arXiv:1804.06824, doi:10.1007/JHEP08(2018)111.

[2] S. Weinberg, The quantum theory of fields, volume 1, Cambridge University Press (2008) 497.

[3] A. V. Kotikov, Differential equations method: New technique for massive Feynman diagrams calculation, Phys. Lett. B254 (1991) 158–164. doi:10.1016/0370-2693(91)90413-K.

[4] E. Remiddi, Differential equations for Feynman graph amplitudes, Nuovo Cim. A110 (1997) 1435–1452. arXiv:hep-th/9711188.

[5] M. Caffo, H. Czyz, S. Laporta, E. Remiddi, Master equations for master amplitudes, Acta Phys. Polon. B29 (1998) 2627–2635. arXiv:hep-th/9807119.

[6] M. Caffo, H. Czyz, S. Laporta, E. Remiddi, The Master differential equations for the two loop sunrise selfmass amplitudes, Nuovo Cim. A111 (1998) 365–389. arXiv:hep-th/9805118.

[7] T. Gehrmann, E. Remiddi, Differential equations for two loop four point functions, Nucl. Phys. B580 (2000) 485–518. arXiv:hep-ph/9912329, doi:10.1016/S0550-3213(00)00223-6.

[8] J. M. Henn, Multiloop integrals in dimensional regularization made simple, Phys. Rev. Lett. 110 (2013) 251601. arXiv:1304.1806, doi:10.1103/PhysRevLett.110.251601.

[9] R. N. Lee, A. V. Smirnov, V. A. Smirnov, Solving differential equations for Feynman integrals by expansions near singular points, JHEP 03 (2018) 008. arXiv:1709.07525, doi:10.1007/JHEP03(2018)008.

[10] X. Liu, Y.-Q. Ma, C.-Y. Wang, A Systematic and Efficient Method to Compute Multi-loop Master Integrals, Phys. Lett. B779 (2018) 353–357. arXiv:1711.09572, doi:10.1016/j.physletb.2018.02.026.

[11] H. Poincaré, Sur les groupes des équations linéaires, , Acta Mathematica 4 (1883) 215.

[12] E. Kummer, Über die Transzendenten, welche aus wiederholten Integrationen rationaler Formeln entstehen , J. Reine Angew. Math. 21 (1840) 74–90, 193–225, 328–371.

[13] N. Nielsen, Der eulersche dilogarithmus und seine verallgemeinerungen, Nova Acta Leopoldina (Halle) 90 123 (1909).

[14] A. B. Goncharov, Geometry of configurations, polylogarithms, and motivic cohomology, Adv. Math. 114 (1995) 197–318.

[15] A. B. Goncharov, Multiple polylogarithms, cyclotomy and modular complexes, Mathematical Research Letters 5 (1998) 497–516.

[16] E. Remiddi, J. A. M. Vermaseren, Harmonic polylogarithms, Int. J. Mod. Phys. A15 (2000) 725–754. arXiv:hep-ph/9905237, doi:10.1142/S0217751X00000367.

[17] J. Vollinga, S. Weinzierl, Numerical evaluation of multiple polylogarithms, Comput. Phys. Commun. 167 (2005) 177. arXiv:hep-ph/0410259, doi:10.1016/j.cpc.2004.12.009.

[18] A. B. Goncharov, M. Spradlin, C. Vergu, A. Volovich, Classical Polylogarithms for Amplitudes and Wilson Loops, Phys. Rev. Lett. 105 (2010) 151605. arXiv:1006.5703, doi:10.1103/PhysRevLett.105.151605.

[19] J. Ablinger, J. Blumlein, C. Schneider, Harmonic Sums and Polylogarithms Generated by Cyclotomic Polynomials, J. Math. Phys. 52 (2011) 102301. arXiv:1105.6063, doi:10.1063/1.3629472.

[20] C. Duhr, Hopf algebras, coproducts and symbols: an application to Higgs boson amplitudes, JHEP 08 (2012) 043. arXiv:1203.0454, doi:10.1007/JHEP08(2012)043.

[21] T. Gehrmann, E. Remiddi, Two loop master integrals for gamma* –> 3 jets: The Nonplanar topologies, Nucl. Phys. B601 (2001) 287–317. arXiv:hep-ph/0101124, doi:10.1016/S0550-3213(01)00074-8.

[22] T. Gehrmann, E. Remiddi, Two loop master integrals for gamma* —> 3 jets: The Planar topologies, Nucl. Phys. B601 (2001) 248–286. arXiv:hep-ph/0008287, doi:10.1016/S0550-3213(01)00057-8.

[23] R. Bonciani, P. Mastrolia, E. Remiddi, Master integrals for the two loop QCD virtual corrections to the forward backward asymmetry, Nucl. Phys. B690 (2004) 138–176. arXiv:hep-ph/0311145, doi:10.1016/j.nuclphysb.2004.04.011.

[24] C. Anastasiou, S. Beerli, S. Bucherer, A. Daleo, Z. Kunszt, Two-loop amplitudes and master integrals for the production of a Higgs boson via a massive quark and a scalar-quark loop, JHEP 01 (2007) 082. arXiv:hep-ph/0611236, doi:10.1088/1126-6708/2007/01/082.

[25] T. Gehrmann, L. Tancredi, E. Weihs, Two-loop master integrals for $q\bar{q} \to VV$: the planar topologies, JHEP 08 (2013) 070. arXiv:1306.6344, doi:10.1007/JHEP08(2013)070.

[26] J. M. Henn, K. Melnikov, V. A. Smirnov, Two-loop planar master integrals for the production of off-shell vector bosons in hadron collisions, JHEP 05 (2014) 090. arXiv:1402.7078, doi:10.1007/JHEP05(2014)090,10.1007/s13130-014-8200-x.

[27] F. Caola, J. M. Henn, K. Melnikov, V. A. Smirnov, Non-planar master integrals for the production of two off-shell vector bosons in collisions of massless partons, JHEP 09 (2014) 043. arXiv:1404.5590, doi:10.1007/JHEP09(2014)043.

[28] T. Gehrmann, A. von Manteuffel, L. Tancredi, E. Weihs, The two-loop master integrals for $q\bar{q} \to VV$, JHEP 06 (2014) 032. arXiv:1404.4853, doi:10.1007/JHEP06(2014)032.

[29] C. G. Papadopoulos, D. Tommasini, C. Wever, The Pentabox Master Integrals with the Simplified Differential Equations approach, JHEP 04 (2016) 078. arXiv:1511.09404, doi:10.1007/JHEP04(2016)078.

[30] T. Gehrmann, J. M. Henn, N. A. Lo Presti, Analytic form of the two-loop planar five-gluon all-plus-helicity amplitude in QCD, Phys. Rev. Lett. 116 (6) (2016) 062001, [Erratum: Phys. Rev. Lett.116,no.18,189903(2016)]. arXiv:1511.05409, doi:10.1103/PhysRevLett.116.189903,10.1103/PhysRevLett.116.062001.

[31] J. M. Henn, A. V. Smirnov, V. A. Smirnov, Analytic results for planar three-loop integrals for massive form factors, JHEP 12 (2016) 144. arXiv:1611.06523, doi:10.1007/JHEP12(2016)144.

[32] R. Bonciani, V. Del Duca, H. Frellesvig, J. M. Henn, F. Moriello, V. A. Smirnov, Two-loop planar master integrals for Higgs$\to$ 3 partons with full heavy-quark mass dependence, JHEP 12 (2016) 096. arXiv:1609.06685, doi:10.1007/JHEP12(2016)096.

[33] M. Becchetti, R. Bonciani, Two-Loop Master Integrals for the Planar QCD Massive Corrections to Di-photon and Di-jet Hadro-production, JHEP 01 (2018) 048. arXiv:1712.02537, doi:10.1007/JHEP01(2018)048.

[34] S. Badger, D. Chicherin, T. Gehrmann, G. Heinrich, J. M. Henn, T. Peraro, P. Wasser, Y. Zhang, S. Zoia, Analytic form of the full two-loop five-gluon all-plus helicity amplitude (2019). arXiv:1905.03733.

[35] F. C. S. Brown, A. Levin, Multiple Elliptic Polylogarithms, ArXiv e-prints (Oct. 2011). arXiv:1110.6917.

[36] J. Broedel, C. Duhr, F. Dulat, L. Tancredi, Elliptic polylogarithms and iterated integrals on elliptic curves I: general formalism (2017). arXiv:1712.07089.

[37] J. Broedel, C. Duhr, F. Dulat, L. Tancredi, Elliptic polylogarithms and iterated integrals on elliptic curves II: an application to the sunrise integral (2017). arXiv:1712.07095.

[38] E. Remiddi, L. Tancredi, An Elliptic Generalization of Multiple Polylogarithms, Nucl. Phys. B925 (2017) 212–251. arXiv:1709.03622, doi:10.1016/j.nuclphysb.2017.10.007.

[39] A. von Manteuffel, L. Tancredi, A non-planar two-loop three-point function beyond multiple polylogarithms, JHEP 06 (2017) 127. arXiv:1701.05905, doi:10.1007/JHEP06(2017)127.

[40] L. Adams, S. Weinzierl, The ε-form of the differential equations for Feynman integrals in the elliptic case (2018). arXiv:1802.05020.

[41] B. Mistlberger, Higgs Boson Production at Hadron Colliders at N^3LO in QCD (2018). arXiv:1802.00833.

[42] K. Hepp, Proof of the bogoliubov-parasiuk theorem on renormalization, Communications in Mathematical Physics 2 (1) (1966) 301–326. doi:10.1007/BF01773358. URL https://doi.org/10.1007/BF01773358

[43] M. Roth, A. Denner, High-energy approximation of one loop Feynman integrals, Nucl. Phys. B479 (1996) 495–514. arXiv:hep-ph/9605420, doi:10.1016/0550-3213(96)00435-X.

[44] T. Binoth, G. Heinrich, An automatized algorithm to compute infrared divergent multiloop integrals, Nucl. Phys. B585 (2000) 741–759. arXiv:hep-ph/0004013, doi:10.1016/S0550-3213(00)00429-6.

[45] G. Heinrich, Sector Decomposition, Int. J. Mod. Phys. A23 (2008) 1457–1486. arXiv:0803.4177, doi:10.1142/S0217751X08040263.

[46] C. Bogner, S. Weinzierl, Resolution of singularities for multi-loop integrals, Comput. Phys. Commun. 178 (2008) 596–610. arXiv:0709.4092, doi:10.1016/j.cpc.2007.11.012.

[47] A. V. Smirnov, M. N. Tentyukov, Feynman Integral Evaluation by a Sector decomposiTion Approach (FIESTA), Comput. Phys. Commun. 180 (2009) 735–746. arXiv:0807.4129, doi:10.1016/j.cpc.2008.11.006.

[48] A. V. Smirnov, V. A. Smirnov, M. Tentyukov, FIESTA 2: Parallelizeable multiloop numerical calculations, Comput. Phys. Commun. 182 (2011) 790–803. arXiv:0912.0158, doi:10.1016/j.cpc.2010.11.025.

[49] A. V. Smirnov, FIESTA 3: cluster-parallelizable multiloop numerical calculations in physical regions, Comput.Phys.Commun. 185 (2014) 2090–2100. arXiv:1312.3186, doi:10.1016/j.cpc.2014.03.015.

[50] J. Gluza, K. Kajda, T. Riemann, V. Yundin, Numerical Evaluation of Tensor Feynman Integrals in Euclidean Kinematics, Eur. Phys. J. C71 (2011) 1516. arXiv:1010.1667, doi:10.1140/epjc/s10052-010-1516-y.

[51] J. Carter, G. Heinrich, SecDec: A general program for sector decomposition, Comput. Phys. Commun. 182 (2011) 1566–1581. arXiv:1011.5493, doi:10.1016/j.cpc.2011.03.026.

[52] S. Borowka, J. Carter, G. Heinrich, Numerical Evaluation of Multi-Loop Integrals for Arbitrary Kinematics with SecDec 2.0, Comput. Phys. Commun. 184 (2013) 396–408. arXiv:1204.4152, doi:10.1016/j.cpc.2012.09.020.

[53] S. Borowka, G. Heinrich, S. P. Jones, M. Kerner, J. Schlenk, T. Zirke, SecDec-3.0: numerical evaluation of multi-scale integrals beyond one loop, Comput. Phys. Commun. 196 (2015) 470–491. arXiv:1502.06595, doi:10.1016/j.cpc.2015.05.022.

[54] S. Borowka, G. Heinrich, S. Jahn, S. P. Jones, M. Kerner, J. Schlenk, T. Zirke, pySecDec: a toolbox for the numerical evaluation of multi-scale integrals, Comput. Phys. Commun. 222 (2018) 313–326. arXiv:1703.09692, doi:10.1016/j.cpc.2017.09.015.

[55] S. Borowka, G. Heinrich, S. Jahn, S. P. Jones, M. Kerner, J. Schlenk, Numerical evaluation of two-loop integrals with pySecDec, Acta Phys. Polon. Supp. 11 (2018) 375. arXiv:1712.05755, doi:10.5506/APhysPolBSupp.11.375.

[56] S. Borowka, T. Hahn, S. Heinemeyer, G. Heinrich, W. Hollik, Momentum-dependent two-loop QCD corrections to the neutral Higgs-boson masses in the MSSM, Eur. Phys. J. C74 (8) (2014) 2994. arXiv:1404.7074, doi:10.1140/epjc/s10052-014-2994-0.

[57] S. Borowka, N. Greiner, G. Heinrich, S. Jones, M. Kerner, J. Schlenk, U. Schubert, T. Zirke, Higgs Boson Pair Production in Gluon Fusion at Next-to-Leading Order with Full Top-Quark Mass Dependence, Phys. Rev. Lett. 117 (1) (2016) 012001, [Erratum: Phys. Rev. Lett.117,no.7,079901(2016)]. arXiv:1604.06447, doi:10.1103/PhysRevLett.117.079901,10.1103/PhysRevLett.117.012001.

[58] S. Borowka, N. Greiner, G. Heinrich, S. P. Jones, M. Kerner, J. Schlenk, T. Zirke, Full top quark mass dependence in Higgs boson pair production at NLO, JHEP 10 (2016) 107. arXiv:1608.04798, doi:10.1007/JHEP10(2016)107.

[59] S. Badger, C. BrÃ¸nnum-Hansen, H. B. Hartanto, T. Peraro, First look at two-loop five-gluon scattering in QCD, Phys. Rev. Lett. 120 (9) (2018) 092001. arXiv:1712.02229, doi:10.1103/PhysRevLett.120.092001.

[60] S. P. Jones, M. Kerner, G. Luisoni, NLO QCD corrections to Higgs boson plus jet production with full top-quark mass dependence (2018). arXiv:1802.00349.

[61] S. Borowka, S. Paßehr, G. Weiglein, Complete two-loop QCD contributions to the lightest Higgs-boson mass in the MSSM with complex parameters (2018). arXiv:1802.09886.

[62] M. Grazzini, G. Heinrich, S. Jones, S. Kallweit, M. Kerner, J. M. Lindert, J. Mazzitelli, Higgs boson pair production at NNLO with top quark mass effects, JHEP 05 (2018) 059. arXiv:1803.02463, doi:10.1007/JHEP05(2018)059.

[63] S. Borowka, C. Duhr, F. Maltoni, D. Pagani, A. Shivaji, X. Zhao, Probing the scalar potential via double Higgs boson production at hadron colliders, JHEP 04 (2019) 016. arXiv:1811.12366, doi:10.1007/JHEP04(2019)016.

[64] G. Heinrich, S. P. Jones, M. Kerner, G. Luisoni, L. Scyboz, Probing the trilinear Higgs boson coupling in di-Higgs production at NLO QCD including parton shower effects, JHEP 06 (2019) 066. arXiv:1903.08137, doi:10.1007/JHEP06(2019)066.

[65] E. Panzer, On hyperlogarithms and Feynman integrals with divergences and many scales, JHEP 03 (2014) 071. arXiv:1401.4361, doi:10.1007/JHEP03(2014)071.

[66] A. von Manteuffel, E. Panzer, R. M. Schabinger, A quasi-finite basis for multi-loop Feynman integrals, JHEP 02 (2015) 120. arXiv:1411.7392, doi:10.1007/JHEP02(2015)120.

[67] R. K. Ellis, W. J. Striling, B. R. Webber, QCD and Collider Physics, Cambridge University Press (1996).

[68] G. Aad, et al., Observation of a new particle in the search for the Standard Model Higgs boson with the ATLAS detector at the LHC, Phys. Lett. B716 (2012) 1–29. arXiv:1207.7214, doi:10.1016/j.physletb.2012.08.020.

[69] S. Chatrchyan, et al., Observation of a New Boson at a Mass of 125 GeV with the CMS Experiment at the LHC, Phys. Lett. B716 (2012) 30–61. arXiv:1207.7235, doi:10.1016/j.physletb.2012.08.021.

[70] C. P. Burgess, G. D. Moore, The Standard Model: A Primer, Cambridge University Press (2007).

[71] S. L. Glashow, Partial Symmetries of Weak Interactions, Nucl. Phys. 22 (1961) 579–588. doi:10.1016/0029-5582(61)90469-2.

[72] S. Weinberg, A model of leptons, Phys. Rev. Lett. 19 (1967) 1264–1266. doi:10.1103/PhysRevLett.19.1264.
URL https://link.aps.org/doi/10.1103/PhysRevLett.19.1264

[73] A. Salam, Weak and Electromagnetic Interactions, Conf. Proc. C680519 (1968) 367–377.

[74] E. D. Bloom, D. H. Coward, H. DeStaebler, J. Drees, G. Miller, L. W. Mo, R. E. Taylor, M. Breidenbach, J. I. Friedman, G. C. Hartmann, H. W. Kendall, High-energy inelastic $e-p$ scattering at 6° and 10°, Phys. Rev. Lett. 23 (1969) 930–934. doi:10.1103/PhysRevLett.23.930.
URL https://link.aps.org/doi/10.1103/PhysRevLett.23.930

[75] M. Breidenbach, J. I. Friedman, H. W. Kendall, E. D. Bloom, D. H. Coward, H. DeStaebler, J. Drees, L. W. Mo, R. E. Taylor, Observed behavior of highly inelastic electron-proton scattering, Phys. Rev. Lett. 23 (1969) 935–939. doi:10.1103/PhysRevLett.23.935.
URL https://link.aps.org/doi/10.1103/PhysRevLett.23.935

[76] J. E. Augustin, A. M. Boyarski, M. Breidenbach, F. Bulos, J. T. Dakin, G. J. Feldman, G. E. Fischer, D. Fryberger, G. Hanson, B. Jean-Marie, R. R. Larsen, V. Lüth, H. L. Lynch, D. Lyon, C. C. Morehouse, J. M. Paterson, M. L. Perl, B. Richter, P. Rapidis, R. F. Schwitters, W. M. Tanenbaum, F. Vannucci, G. S. Abrams, D. Briggs, W. Chinowsky, C. E. Friedberg, G. Goldhaber, R. J. Hollebeek, J. A. Kadyk, B. Lulu, F. Pierre, G. H. Trilling, J. S. Whitaker, J. Wiss, J. E. Zipse, Discovery of a narrow resonance in e^+e^- annihilation, Phys. Rev. Lett. 33 (1974) 1406–1408. doi:10.1103/PhysRevLett.33.1406.
URL https://link.aps.org/doi/10.1103/PhysRevLett.33.1406

[77] J. J. Aubert, U. Becker, P. J. Biggs, J. Burger, M. Chen, G. Everhart, P. Goldhagen, J. Leong, T. McCorriston, T. G. Rhoades, M. Rohde, S. C. C. Ting, S. L. Wu, Y. Y. Lee, Experimental observation of a heavy particle j, Phys. Rev. Lett. 33 (1974) 1404–1406. doi:10.1103/PhysRevLett.33.1404.
URL https://link.aps.org/doi/10.1103/PhysRevLett.33.1404

[78] S. W. Herb, D. C. Hom, L. M. Lederman, J. C. Sens, H. D. Snyder, J. K. Yoh, J. A. Appel, B. C. Brown, C. N. Brown, W. R. Innes, K. Ueno, T. Yamanouchi, A. S. Ito, H. Jöstlein, D. M. Kaplan, R. D. Kephart, Observation of a dimuon resonance at 9.5 gev in 400-gev proton-nucleus collisions, Phys. Rev. Lett. 39 (1977) 252–255. doi:10.1103/PhysRevLett.39.252.
URL https://link.aps.org/doi/10.1103/PhysRevLett.39.252

[79] G. Arnison, et al., Experimental Observation of Lepton Pairs of Invariant Mass Around 95-GeV/c**2 at the CERN SPS Collider, Phys. Lett. B126 (1983) 398–410, [,7.55(1983)]. doi:10.1016/0370-2693(83)90188-0.

[80] F. Abe, et al., Observation of top quark production in $\bar{p}p$ collisions, Phys. Rev. Lett. 74 (1995) 2626–2631. arXiv:hep-ex/9503002, doi:10.1103/PhysRevLett.74.2626.

[81] S. Abachi, et al., Search for high mass top quark production in $p\bar{p}$ collisions at $\sqrt{s} = 1.8$ TeV, Phys. Rev. Lett. 74 (1995) 2422–2426. arXiv:hep-ex/9411001, doi:10.1103/PhysRevLett.74.2422.

[82] K. Kodama, et al., Observation of tau neutrino interactions, Phys. Lett. B504 (2001) 218–224. arXiv:hep-ex/0012035, doi:10.1016/S0370-2693(01)00307-0.

[83] F. Englert, R. Brout, Broken symmetry and the mass of gauge vector mesons, Phys. Rev. Lett. 13 (1964) 321–323. doi:10.1103/PhysRevLett.13.321.
URL https://link.aps.org/doi/10.1103/PhysRevLett.13.321

[84] P. W. Higgs, Broken symmetries, massless particles and gauge fields, Phys. Lett. 12 (1964) 132–133. doi:10.1016/0031-9163(64)91136-9.

[85] P. W. Higgs, Broken symmetries and the masses of gauge bosons, Phys. Rev. Lett. 13 (1964) 508–509. doi:10.1103/PhysRevLett.13.508.
URL https://link.aps.org/doi/10.1103/PhysRevLett.13.508

[86] P. W. Higgs, Spontaneous symmetry breakdown without massless bosons, Phys. Rev. 145 (1966) 1156–1163. doi:10.1103/PhysRev.145.1156.
URL https://link.aps.org/doi/10.1103/PhysRev.145.1156

[87] G. S. Guralnik, C. R. Hagen, T. W. B. Kibble, Global conservation laws and massless particles, Phys. Rev. Lett. 13 (1964) 585–587. doi:10.1103/PhysRevLett.13.585.
URL https://link.aps.org/doi/10.1103/PhysRevLett.13.585

[88] K. Riley, M. Hobson, S. Bence, Mathematical Methods for Physics and Engineering, Cambridge University Press (2002).

[89] M. Viazovska, The sphere packing problem in dimension 8, arXiv e-prints (2016) arXiv:1603.04246arXiv:1603.04246.

[90] K. Knopp, Theory of Functions Part 1, Dover (1945).

[91] E. G. Phillips, Functions of a Complex Variable with Applications, Oliver and Boyd (1951).

[92] T. Gehrmann, S. Guns, D. Kara, The rare decay $H \to Z\gamma$ in perturbative QCD, JHEP 09 (2015) 038. arXiv:1505.00561, doi:10.1007/JHEP09(2015)038.

[93] M. E. Peskin, D. V. Schroeder, An Introduction to Field Theory, Westview Press (1995).

[94] V. A. Smirnov, Feynman Integral Calculus, Springer (2006).

[95] G. 't Hooft, M. J. G. Veltman, Regularization and Renormalization of Gauge Fields, Nucl. Phys. B44 (1972) 189–213. doi:10.1016/0550-3213(72)90279-9.

[96] C. G. Bollini, J. J. Giambiagi, Dimensional Renormalization: The Number of Dimensions as a Regularizing Parameter, Nuovo Cim. B12 (1972) 20–26. doi:10.1007/BF02895558.

[97] J. W. York, Jr., Role of conformal three geometry in the dynamics of gravitation, Phys. Rev. Lett. 28 (1972) 1082–1085. doi:10.1103/PhysRevLett.28.1082.

[98] J. F. Ashmore, A Method of Gauge Invariant Regularization, Lett. Nuovo Cim. 4 (1972) 289–290. doi:10.1007/BF02824407.

[99] G. Passarino, M. J. G. Veltman, One Loop Corrections for e+ e- Annihilation Into mu+ mu- in the Weinberg Model, Nucl. Phys. B160 (1979) 151–207. doi:10.1016/0550-3213(79)90234-7.

[100] Z. Kunszt, A. Signer, Z. Trocsanyi, One loop helicity amplitudes for all 2 —> 2 processes in QCD and N=1 supersymmetric Yang-Mills theory, Nucl. Phys. B411 (1994) 397–442. arXiv:hep-ph/9305239, doi:10.1016/0550-3213(94)90456-1.

[101] A. Signer, D. Stockinger, Using Dimensional Reduction for Hadronic Collisions, Nucl. Phys. B808 (2009) 88–120. arXiv:0807.4424, doi:10.1016/j.nuclphysb.2008.09.016.

[102] K. Aomoto, M. Kita, Theory of Hypergeometric Functions, Springer (2011).

[103] C. Bogner, S. Weinzierl, Feynman graph polynomials, Int. J. Mod. Phys. A25 (2010) 2585–2618. arXiv:1002.3458, doi:10.1142/S0217751X10049438.

[104] M. Argeri, P. Mastrolia, Feynman Diagrams and Differential Equations, Int. J. Mod. Phys. A22 (2007) 4375–4436. arXiv:0707.4037, doi:10.1142/S0217751X07037147.

[105] S. Laporta, High precision calculation of multiloop Feynman integrals by difference equations, Int. J. Mod. Phys. A15 (2000) 5087–5159. arXiv:hep-ph/0102033, doi:10.1016/S0217-751X(00)00215-7,10.1142/S0217751X00002157.

[106] A. V. Smirnov, Algorithm FIRE – Feynman Integral REduction, JHEP 10 (2008) 107. arXiv:0807.3243, doi:10.1088/1126-6708/2008/10/107.

[107] A. von Manteuffel, C. Studerus, Reduze 2 - Distributed Feynman Integral Reduction (2012). arXiv:1201.4330.

[108] S. G. Gorishnii, S. A. Larin, L. R. Surguladze, F. V. Tkachov, Mincer: Program for Multiloop Calculations in Quantum Field Theory for the Schoonschip System, Comput. Phys. Commun. 55 (1989) 381–408. doi:10.1016/0010-4655(89)90134-3.

[109] C. Anastasiou, A. Lazopoulos, Automatic integral reduction for higher order perturbative calculations, JHEP 07 (2004) 046. arXiv:hep-ph/0404258, doi:10.1088/1126-6708/2004/07/046.

[110] F. V. Tkachov, A Theorem on Analytical Calculability of Four Loop Renormalization Group Functions, Phys. Lett. 100B (1981) 65–68. doi:10.1016/0370-2693(81)90288-4.

[111] F. K. Chetyrkin, Integration by parts: The algorithm to calculate beta functions in 4 loops, Nucl.Phys. B 192 (1) (1981) 159–204. doi:10.1016/0550-3213(81)90199-1.

[112] L. D. Landau, On analytic properties of vertex parts in quantum field theory, Nucl. Phys. 13 (1959) 181–192. doi:10.1016/0029-5582(59)90154-3.

[113] R. E. Cutkosky, Singularities and discontinuities of Feynman amplitudes, J. Math. Phys. 1 (1960) 429–433. doi:10.1063/1.1703676.

[114] R. Eden, D. Landshoff, P. Olive, J. Polkinghorne, The analytic S-matrix, International Serie in Pure and Applied Physics, McGraw-Hill (1966).

[115] C. Itzykson, J. B. Zuber, Quantum Field Theory, International Serie in Pure and Applied Physics, McGraw-Hill (1980).

[116] S. Coleman, R. E. Norton, Singularities in the Physical Region, Nuovo Cim. 38 (1965) 438–442.

[117] C. W. Bauer, A. Frink, R. Kreckel, Introduction to the GiNaC framework for symbolic computation within the C++ programming language, J. Symb. Comput. 33 (2000) 1. arXiv:cs/0004015.

[118] Mathematica, Copyright by Wolfram Research.

[119] J. A. M. Vermaseren, New features of FORM (2000). arXiv:math-ph/0010025.

[120] J. Kuipers, T. Ueda, J. A. M. Vermaseren, Code Optimization in FORM, Comput. Phys. Commun. 189 (2015) 1–19. arXiv:1310.7007, doi:10.1016/j.cpc.2014.08.008.

[121] T. Kaneko, T. Ueda, A Geometric method of sector decomposition, Comput.Phys.Commun. 181 (2010) 1352–1361. arXiv:0908.2897, doi:10.1016/j.cpc.2010.04.001.

[122] T. Kaneko, T. Ueda, Sector Decomposition Via Computational Geometry, PoS ACAT2010 (2010) 082. arXiv:1004.5490.

[123] H. Cheng, T. Wu, Expanding Protons: Scattering at High Energies, The MIT Press, 1987.

[124] V. A. Smirnov, Feynman integral calculus, Springer, 2006.

[125] R. Bonciani, P. Mastrolia, E. Remiddi, Vertex diagrams for the QED form-factors at the two loop level, Nucl. Phys. B661 (2003) 289–343, [Erratum: Nucl. Phys.B702,359(2004)]. arXiv:hep-ph/0301170, doi:10.1016/S0550-3213(03)00299-2,10.1016/j.nuclphysb.2004.08.009.

[126] A. Primo, L. Tancredi, On the maximal cut of Feynman integrals and the solution of their differential equations, Nucl. Phys. B916 (2017) 94–116. arXiv:1610.08397, doi:10.1016/j.nuclphysb.2016.12.021.

[127] K. Melnikov, L. Tancredi, C. Wever, Two-loop amplitudes for $qg \to Hq$ and $q\bar{q} \to Hg$ mediated by a nearly massless quark, Phys. Rev. D95 (5) (2017) 054012. arXiv:1702.00426, doi:10.1103/PhysRevD.95.054012.

[128] G. Passarino, S. Uccirati, Algebraic numerical evaluation of Feynman diagrams: Two loop selfenergies, Nucl. Phys. B629 (2002) 97–187. arXiv:hep-ph/0112004, doi:10.1016/S0550-3213(02)00138-4.

[129] D. T. Nhung, L. D. Ninh, D0C : A code to calculate scalar one-loop four-point integrals with complex masses, Comput. Phys. Commun. 180 (2009) 2258–2267. arXiv:0902.0325, doi:10.1016/j.cpc.2009.07.012.

[130] G. Passarino, Peaks and cusps: anomalous thresholds and LHC physics (2018). arXiv:1807.00503.

[131] G. P. Lepage, A New Algorithm for Adaptive Multidimensional Integration, J. Comput. Phys. 27 (1978) 192. doi:10.1016/0021-9991(78)90004-9.

[132] P. Mastrolia, M. Passera, A. Primo, U. Schubert, Master integrals for the NNLO virtual corrections to μe scattering in QED: the planar graphs, JHEP 11 (2017) 198. arXiv:1709.07435, doi:10.1007/JHEP11(2017)198.

List of Figures

16 | Acknowledgements

First and foremost, I would like to express my deep gratitude to Prof. Dr. Thomas Gehrmann and Dr. Sophia Borowka, my research supervisors, for their patient guidance, enlightening discussions, motivational ideas and constant support. It has been a great honour to work with them. In particular, I am grateful to Thomas Gehrmann for his relentlessly brilliant physics and mathematical insight, willingness to share it at any time, and the combination of enthusiasm for my work and patience with my manner of carrying it out that allowed this project to develop. I am especially grateful to Sophia Borowka for sharing her outstanding command of computer programming, her wisdom concerning all manners of attacking loop integrals and her gift for always being able to see something positive in my work. I wish to thank both for their outstanding organisation and dedication both to my academic and personal development, and for providing me with many great opportunities to travel to conferences and workshops.

I gratefully acknowledge several constructive conversations with Dr. Erik Panzer, Dr. Andreas von Manteuffel, Dr. Amedeo Primo and Dr. Dominik Kara during the course of this project. I would like to particularly think Dominik Kara for his support with understanding the intricacies of the Higgs+jet process and Amedeo Primo for his support with divergent Feynman integrals.

I would also like to extend my thanks to Dr Roland Bernet for frequently sorting out my various computing problems in a quick and timely manner.

Furthermore, I wish to thank all current and former PhD students and PostDocs of the theory group in Zürich, especially Federico Buccioni, Luca Buonocore, Leandro Cieri, Simone Devoto, Nico Greiner, Marius Höfer, Tomas Jezo, Dominik Kara, Alexander Karlberg, Matthias Kerner, Jean-Nicolas Lang, Jonas Lindert, Javier Mazzitelli, Jan Nieheus, Amedeo Primo, Andrea Visconti, Marius Wiesmann, Jeong Yeon Yook and Max Zoller for creating such a wonderful atmosphere at the institute, and for all the great times playing board games, at muscle pump, swimming, skiing and playing tennis. I want to thank my office mates Nico Greiner, Marius Höfer, and Matthias Kerner for interesting discussions and creating a dedicated and productive working environment.

To those amongst the aforementioned who gave up their time to help with proofreading this thesis, I am also exceedingly grateful.

I am also very grateful for the variety of literature, knowledgeable colleagues, contemporary equipment, fast and effective IT support provided by the institute. Moreover,

I would like to thank all people working at the institute for contributing to such a friendly atmosphere, especially the current and former secretaries of the theory department Regina Schmid, Carmelina Genovese, Gaby Aeppli and Monika Röllin.

No list of acknowledgements would ever be complete without including my heartfelt thanks to my best friends, Peter Stevens, Bethanie Parker, Katherine Hardwick, David Heavyside, Rowan von Spreckelsen and Rachel Brodie-Browne, who have provided me with the best memories, and whose company always keeps me excited for the future. My Father and Mother cultivated my mind, taught me the value of discipline and integrity and continuously supported my education from the very beginning, and my sisters Teresa, Catherine and Sarah helped my grow up in so many ways, as well as making the process incredibly fun. I cannot ever thank them enough for all that they have given me. The same is equally true of my wonderful girlfriend Martina, who taught me to feel the meanings of all the most superlative words in the English language in our own personal way, and rather than repeating them all here, I will simply give her my final and eternal thanks.